MODEL PREDICTIVE CONTROL (MPC) ALGORITHM FOR TIP-JET REACTION DRIVE SYSTEMS

A Thesis
Presented to
The Academic Faculty

by

Brian Kestner

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
December 2009

MODEL PREDICTIVE CONTROL (MPC) ALGORITHM FOR TIP-JET REACTION DRIVE SYSTEMS

Approved by:

Prof. Dimitri N. Mavris, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Jimmy Tai
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Tim Healy
Gas Turbine Combustion Engineering
General Electric Energy

Mr. Randy Rosson
Gas Turbine Performance Methods
General Electric Energy

Prof. Brian German
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: November 13, 2009

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Dimitri Mavris, for giving me the opportunity to join the ASDL and pursue my PhD. In addition to providing me with this opportunity, his philosophy towards research was most inspiring.

I would like to especially thank Dr. Jimmy Tai for all the support, advice, and help over the last five years. The formulation of this dissertation came out of our weekly meetings. I could not have completed this dissertation without his help.

I would like to thank Dr. Tim Healy and Mr. Randy Rosson. The feedback and help provided was always most enlightening, even though it usually took me awhile to realize it. I would to thank Professor Brian German for always keeping an eye on me.

To the love of my life, Gina Martell, thank you for being there through all the good and the bad. I know it has not always been easy over the last year but I am forever grateful for the love and support. Coming home to your smile always cheered me up when I was stressed out. Thanks to my mom, dad, sister, and brother whose words of encouragement and patience allowed me keep my focus when the going got tough.

Also I would like to say thanks to all my friends both here in Atlanta or back home in Milwaukee for providing encouragement and the necessary distractions to keep me sane over this five year odyssey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

CEPR	Core Engine Pressure Ratio
CRW	Canard Rotor Wing
DLL	Dynamic Link Library
EKF	Extended Kalman Filter
EPR	Engine Pressure Ratio
FCMMPC	Feasible Cooperative Multiplexed Model Predictive Control
HP	High Pressure
HPC	High Pressure Compressor
HPT	High Pressure Turbine
JSF	Joint Strike Fighter
LQG	Linear Quadratic Gaussian
LP	Low Pressure
LPT	Low Pressure Turbine
LTR	Loop Transfer Recovery
MBC	Model Based Control
MIMO	Multiple-In, Multiple-Out
MPC	Model Predictive Control
M&S	Modeling and Simulation
NPSS	National Propulsion System Simulation
PI	Proportional-Integral
RGA	Relative Gain Array
SISO	Single-In, Single-Out
STOVL	Short Takeoff and Vertical Landing

SVD	Singular Value Decomposition
TSFC	Thrust Specific Fuel Consumption
VSV	Variable Stator Vanes
VTOL	Vertical Takeoff and Landing

SUMMARY

Modern technologies coupled with advanced research have allowed model predictive control (MPC) to be applied to new and often experimental systems. The purpose of this research is to develop a model predictive control algorithm for tip-jet reaction drive system. This system's faster dynamics require an extremely short sampling rate, on the order of 20ms, and its slower dynamics require a longer prediction horizon. This coupled with the fact that the tip-jet reaction drive system has multiple control inputs makes the integration of an online MPC algorithm challenging. In order to apply a model predictive control to the system in question, an algorithm is proposed that combines multiplexed inputs and a feasible cooperative MPC algorithm.

In the proposed algorithm, it is hypothesized that the computational burden will be reduced from approximately $H_p(N_u + N_x)^3$ to $pH_p(N_x+1)^3$ while maintaining control performance similar to that of a centralized MPC algorithm. To capture the performance capability of the proposed controller, a comparison its performance to that of a multivariable proportional-integral (PI) controller and a centralized MPC is executed. The sensitivity of the proposed MPC to various design variables is also explored. In terms of bandwidth, interactions, and disturbance rejection, the proposed MPC was very similar to that of a centralized MPC or PI controller. Additionally in regards to sensitivity to modeling error, there is not a noticeable difference between the two MPC controllers. Although the constraints are handled adequately for the proposed controller, adjustments can be made in the design and sizing process to improve the constraint handling, so that it is more comparable to that of the centralized MPC. Given these observations, the hypothesis of the dissertation has been confirmed. The proposed MPC

does in fact reduce computational burden while maintaining close to centralized MPC performance.

CHAPTER 1

INTRODUCTION

Modern technologies coupled with advanced research have allowed model predictive control (MPC) to be applied to new and often experimental systems. The purpose of this research is to develop a model predictive control algorithm for tip-jet reaction drive system. For this system the faster dynamics of the system require an extremely short sampling rate (on the order of 20ms) and the slower dynamics require a longer prediction horizon. This coupled with the fact that the tip-jet reaction drive system has multiple control inputs makes the integration of an online MPC algorithm for the tip-jet reaction drive challenging.

In Chapter 2 of this dissertation, the tip-jet reaction drive system is introduced and various different multivariable control schemes that may be used to control it are discussed. From this analysis, the constrained optimized approach of an MPC was identified as an attractive option. In Chapter 3, the MPC algorithm is introduced in more detail as well as a literature review of MPC on some similar systems to a tip-jet reaction drive system is performed. Chapter 4 discusses in detail the challenges in developing a MPC for a tip-jet reaction drive system.

To address these challenges, this dissertation hypothesizes that combining a multiplexing and feasible cooperative MPC scheme thus reducing the computational burden from approximately $H_p(N_u + N_x)^3$ to $pH_p(N_x+1)^3$ while maintaining control performance similar to that of a centralized MPC algorithm. The process of moving from the technical challenges of developing a tip-jet reaction drive MPC to the proposed MPC is outlined in Chapter 4. In Chapter 5, a detailed discussion of the mathematical background of the constrained optimization for the tip-jet reaction drive system is performed. The analysis in this chapter allows for a quantification of the computational burden reduction gained by using the proposed MPC algorithm.

To capture the performance capability of the proposed controller, a comparison of the performance of the proposed controller to that of a couple baseline controllers is required. The baseline controllers were chosen to be a multivariable proportional-integral (PI) controller and a centralized MPC. The multivariable PI controller and centralized MPC for the tip-jet reaction are developed in Chapters 6 and 7, respectively. The proposed MPC is developed is Chapter 8. Using the results from these chapters, the performance of each of the controllers is compared in Chapter 9. From the results of the performance comparison in Chapter 9 and the matrix exploration in Chapter 5, the hypothesis of the dissertation can be confirmed. Chapter 10 provides a concluding summary and provides recommendations for future work.

CHAPTER 2

PROBLEM STATEMENT

Recently there has been interest in a vertical takeoff and landing (VTOL) rotorcraft that can approach 450 knots. For this application to be successful it must perform similar to a helicopter at low speeds and during hover while capturing the performance qualities of a fixed wing aircraft at higher cruise speeds. In the early 1990s, NASA funded an industry wide study to analyze different concepts to meet this need (TALBOT 1991). Out of this research, the rotor/wing, or tip-jet reaction drive, concept was identified as a potential attractive option.

2.1 **Tip-Jet Reaction Drive System**

The tip-jet reaction drive system combines in a single configuration the capability to perform both like a helicopter at low speeds and hover and like a fixed wing aircraft at high speeds. To enable these different capabilities, the tip-jet reaction drive system has multiple modes of operation, as shown in Figure 1. In hover and VTOL operation, the cold, warm, or hot engine air is ducted through the rotor. This air is then combusted and accelerated using a tip-jet combustor and exhaust nozzle thus generating the necessary lift and forward thrust. During low-speed flight mode, the rotor is no longer powered, but rather is in autorotation. As the aircraft transitions from low-speed to high-speed flight mode, the lifting is initially done primarily by the rotor, but is increasingly offloaded to the wing as the vehicle gains speed. In both low and high-speed flight mode, the engine is the sole provider of forward thrust.

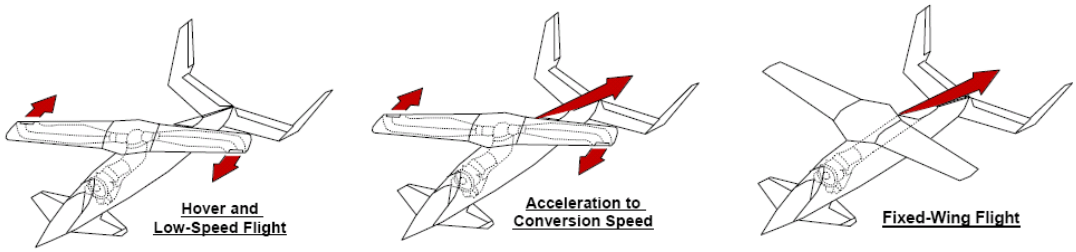

Figure 1: Tip-Jet Reaction Drive Modes of Operation (MITCHELL and VOGEL 2003)

There are a few different configurations of tip-jet reaction drive system with the main difference being the temperature of the engine air supplied to the rotor. Figure 2 and Figure 3 below show a cold and hot/warm cycle tip-jet reaction drive system, respectively. The cold cycle ducts cooler compressor or fan discharge air into the duct rotor; whereas the hot/warm cycle ducts the hotter engine exhaust air into the duct rotor. The cold cycle may require more than one engine because only the core bypass air is ducted to the rotor. A hot cycle would be typically seen with a turbojet engine where all the air supplied to the rotor comes from the engine core. In contrast, a warm cycle would be seen with a mixed flow turbofan engine, where the hot core air gets mixed with the much cooler core bypass air before being supplied to the rotor. Boeing developed an aircraft named the Dragonfly, which was powered by a warm cycle reaction drive system (MITCHELL and VOGEL 2003).

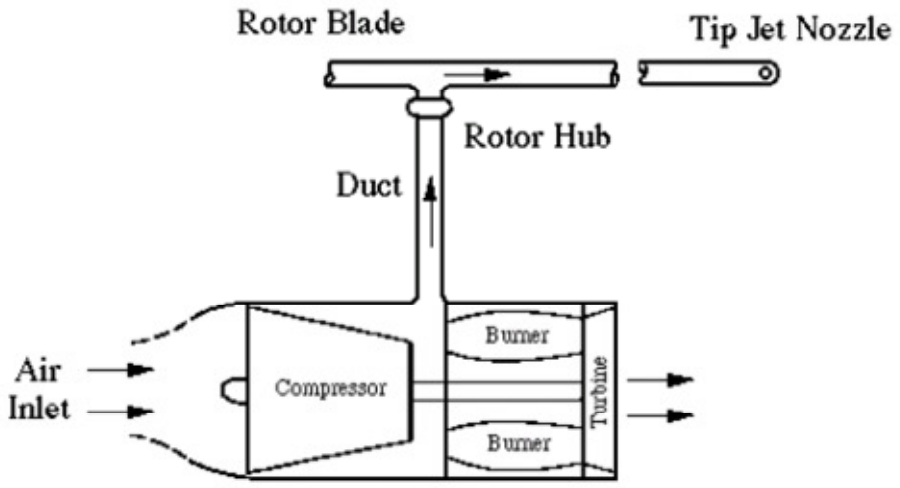

Figure 2: "Cold" Cycle Tip-Jet Reaction Drive System (MAVRIS, TAI and SCHRAGE 1994)

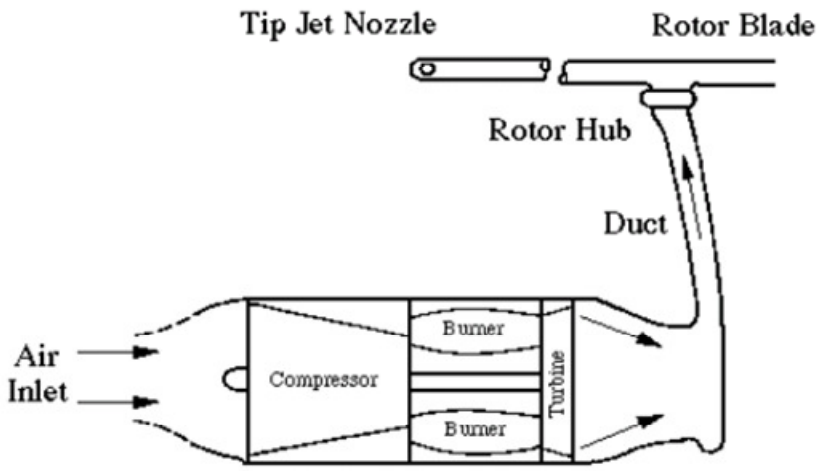

Figure 3: "Hot" or "Warm" Cycle Tip-Jet Reaction Drive System(MAVRIS, TAI and SCHRAGE 1994)

For safe and stable operation of the tip-jet reaction drive system, the control system must be able to provide the necessary thrust or lift required depending on the mode of operation while accounting for the interactions between the engines and the rotor. Depending on the mode of operation, the interactions between the engines and rotor (the subsystems) may change. In hover mode, when the bypass air of the engines is ducted into the rotor, the system is highly interactive. However, in flight mode, the interaction between the engines and the rotor is minimal. In addition to safety and stability, optimization of the aircraft and propulsion system must also be taken into account. Typically, the most optimal performance setting is minimal fuel burn; however, there may be certain situations, such as a damaged aircraft, where minimum fuel burn might not be the optimal setting. One last criterion for safe control of the system is to update the control at a rate that is consistent with the fastest significant dynamics of the system. For the tip-jet reaction drive system the dynamics of the system are relatively fast and require a fairly fast control update rate of approximately 20ms. This leads to the problem statement of this dissertation:

Develop control laws to ensure safe, stable, and optimal operation for the tip-jet reaction driven system.

Although the research in the subsequent chapters can be performed on any tip-jet reaction drive configuration, the research will focus on a cold cycle using two engines powering the rotor. Any further reference to the tip-jet reaction drive system will explicitly mean this configuration. The first step in developing the control laws for the tip-jet reaction drive system is to define the system that is to be controlled. Due to varying degradation, the presence of faults, and manufacturing variation, the performance of the two engines may not be identical and the corresponding effects of these differences need to be quantified. Therefore the system model must include both engines and the tip-jet driven rotor as illustrated in Figure 4.

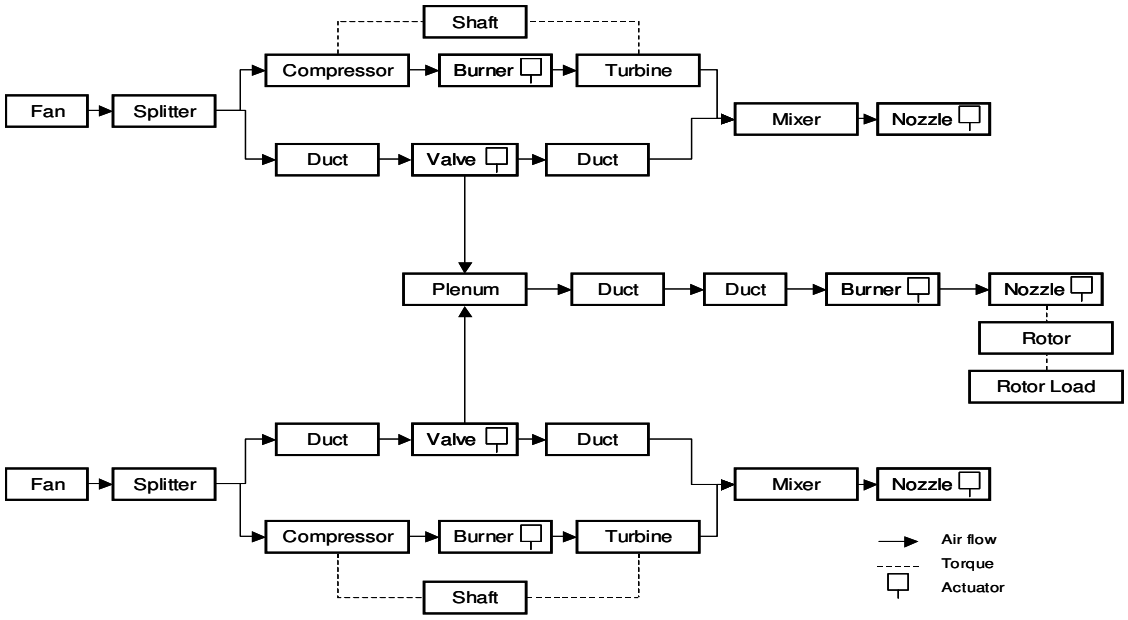

Figure 4: Tip-Jet Reaction Drive System Schematic

The next step in developing control laws is to define which target variables will be controlled, i.e., to put it simply, the purpose of the controller. Depending on the system, the control targets can range from performance parameters, state variables, or sensed variables. For the tip-jet reaction drive system the target control parameters are

the thrust for each of the two engines and the lift/thrust of the rotor system. In addition to controlling to target parameters, a control system should be designed to ensure that certain performance parameters or constraints are not exceeded. These constraints can be physical boundaries such as actuator position, life limiting factors such as turbine inlet temperature or rotor shaft speed, or operability limits such as compressor stall margin. The developer of the control system should design the control such that the system tracks the target parameters while ensuring that the violation of constraints is minimal at worst.

After the target control parameters of the system are defined, the variables used to control the system need to be defined. Figure 4 shows the actuators (# of actuators) that are available to control the tip-jet system: engine fuel metering valves (2), engine exhaust nozzle (2), engine diverter valve (2), rotor fuel metering valve (4), and rotor exhaust nozzle (4). In general, the fuel metering valves are used to control thrust or lift while the exhaust valves are used to ensure that operability or performance limits are not exceeded. If there were no interactions between all the inputs and outputs, each controller could be a simple single-in, single-out (SISO) controller. However, it is clear that the system is highly coupled with potentially strong interactions and cross coupling between the inputs and outputs. The fact that the Dragonfly crashed due to unaccounted for cross coupling indicates that this is not a trivial issue (MCKENNA 2007). Therefore it is imperative that the control system developed for the tip-jet reaction drive system is able to account for all the interactions and to decouple them.

2.2 **Multivariable Control Methods**

Control systems developed to capture and decouple the system interactions are commonly referred to as multiple-in, multiple-out (MIMO) or multivariable controllers. In addition to the interactions, Maciejowski and Dadd defined criteria that are important for the design of a multivariable control system (MACIEJOWSKI 1989)(DADD, SUTTON and GRIEG 1995). The control system must be able to accommodate the

system dynamics across the flight envelope, for varying throttle settings, and for different modes of operation. Additionally, any new controller designed should be able to provide a response comparable with or better than any existing controllers. The control system must be able reject disturbances that have dynamics which are as fast as other control dynamics. As mentioned above the control system must ensure that all system structural and physical limits along with all actuator physical and rate limits must be avoided. And lastly, the control system should be robust enough to handle other variability such as manufacturing, component degradation, and component asymmetry. A successful multivariable control will ensure that all these design criteria are met.

In describing the subsequent multivariable control methodologies, a common set of notation can be used to describe them. In general most control systems are considered reference tracking. The system that is being controlled can be represented by G(s). The purpose of the control is to adjust the system inputs, û(s), to meet a reference value, $\hat{r}(s)$. Depending on the type of control, the reference value is compared to the system output, $\hat{y}(s)$, or an estimation of system output, $\hat{z}(s)$ and an error, $\hat{e}(s)$, is calculated. Given this error, the system inputs are adjusted accordingly via some sort of gain, K(s). The control must be robust enough to maintain acceptable performance in the presence of measurement errors, $\hat{m}(s)$, and disturbances on the system, $\hat{d}(s)$.

2.2.1 Model Based and Non-Model Based Control

Over the last 30 years many different multivariable control methodologies able to handle the aforementioned design criteria have been developed. They can generally be broken down into two different categories: non-model based and model based. Model based methods have an onboard model used to estimate unmeasured parameters; allowing the controller to directly control to these estimates. In contrast non-model based methods are limited to controlling the system using the available measured parameters.

Non-model based control architecture is shown in Figure 5. The first category to look at is the non-model based methods. The main advantage of non-model based methods (when compared with model based methods) is they are very computationally efficient (i.e., fast) and have been proven effective over the last 30 years (JAW and GARG 2005). The main drawback of non-model based method is that the control parameters are limited only to the parameters that are sensed. This factor may have an adverse effect on the performance of the system and constraint handling. Many performance parameters used in control development, such as thrust, cannot be directly measured. To account for this issue, a relationship between these parameters and sensed variables is quantified. However, as the system performance changes over time due to factors such as degradation, the relationship may change and result in less than optimal or margined system performance. Often constraints, such as stall margin or turbine inlet temperature cannot be measured, and if these constraints are violated, the control system cannot adjust accordingly. The indirect control of the system along with the inability to account for certain system constraints inherent in non-model based methods are severe limiting factors for safe, stable, optimal control of a system.

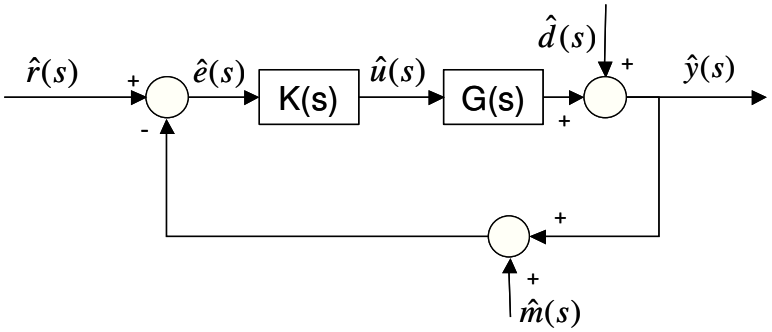

Figure 5: Non-model Based Control Architecture

Compared with non-model based methods, model based control (MBC) methods offer the advantage of direct controlling to both unmeasured and measured parameters in addition to direct handling of constraints. In MBC methods, an onboard system model calculates all the performance parameters critical to optimal system performance. As the

system degrades, the onboard model can capture the changes to the system and adjust accordingly. The onboard model also quantifies safety parameters, such as stall margin or turbine inlet temperature, that constrain the operation of the system allowing the control to adjust when these usually immeasurable constraints are reached. This ensures that the system operates at a safer point. Figure 6 below represents typical model based control architecture. In this figure, the state of the system is assumed not to be measured and requires the use of a state estimator, such as Kalman or particle filters (KOBAYASHI, SIMON and LITT 2005)(ARULAMPALAM, et al. 2002), to estimate the magnitude of the state from the measured output. If the state can be measured, the state estimator is not a required. In regards to aerospace applications, model based controls have been demonstrated on propulsion systems of the F-22 and the Joint Strike Fighter (JSF)(JAW and GARG 2005) (VOLPONI 2008). Despite its many advantages, MBC has two major drawbacks. These are the increased complexity and computational burden of the control as a result of the requirement of a model that matches actual engine performance.

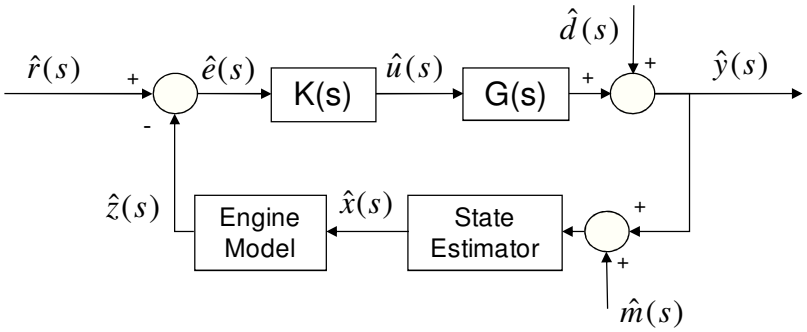

Figure 6: Model Based Control Architecture

2.2.2 Multivariable Control Methodologies

Four different multivariable control methodologies have been identified as potential options for controlling the tip-jet reaction drive system. The first is a simple proportional-integral (PI) controller. The second is the robust control technique known

as H-infinity. The last two are both optimal type controllers: linear quadratic Gaussian (LQG) and model predictive control. Each control methodology will have a brief introduction to the control architecture, a short discussion of the theory behind it, and lastly a short summary of the pros and cons.

2.2.2.1 Proportional-Integral (PI)

Of all the multivariable control methodologies explored in the feasibility study, the PI controller is the most simple. The PI controller can be used on both the non-model based or model based architectures as shown in Figure 5 and Figure 6. There are two components to a PI controller as shown in equation 1. First there is a proportional gain, K_p, which is a gain applied to the current sampled reference tracking error. The second component is the integral gain, K_I, which is a gain applied to the integral of the reference tracking error over the last sampling interval. Although the integral gain slows down the response, it ensures that there is zero steady-state error tracking.

$$\hat{u}(s) = K_p \hat{e}(s) + K_I \int \hat{e}(s) dt \qquad [1]$$

The Edmunds algorithm developed in the late 1970's has provided the foundation for recent multivariable PI engine control (EDMUNDS 1979). This methodology derives a set of PI control gains based upon a target open loop response of the system. Polley provided an extension to this methodology to ensure similar system response over the flight envelope (POLLEY, ADIBHATLA and HOFFMAN 1989). PI controllers have been proven reliable for years and have been used on numerous different aerospace applications. PI controllers have even been studied for use on the propulsion controller for short takeoff vertical landing (STOVL) applications (ADIBHATLA 1993).

A drawback to PI controllers is that controller is designed deterministically and may not be robust enough to account for various uncertainties in the process, the process model, or the available sensors. To ensure that the control is robust enough, this may make the design process fairly iterative and time. Because of the simple control structure

(i.e. low order control), even after multiple iteration, the control may not be robust enough. Other methods such as H-infinity and linear quadratic Gaussian (LQG) ensure a more robust control at the expense of less simple control (WATTS and GARG 1995).

2.2.2.2 H Infinity

Figure 7 below shows a potential H-infinity control architecture. Functionally speaking, an H-infinity control is virtually identical to the PI controller except that the controller is of a much higher order. The output of blocks of W1 and W2 shown in Figure 7 are used only used in the sizing of the controller and are not used functionally. Both model based and non-model based control can be used with an H-infinity controller.

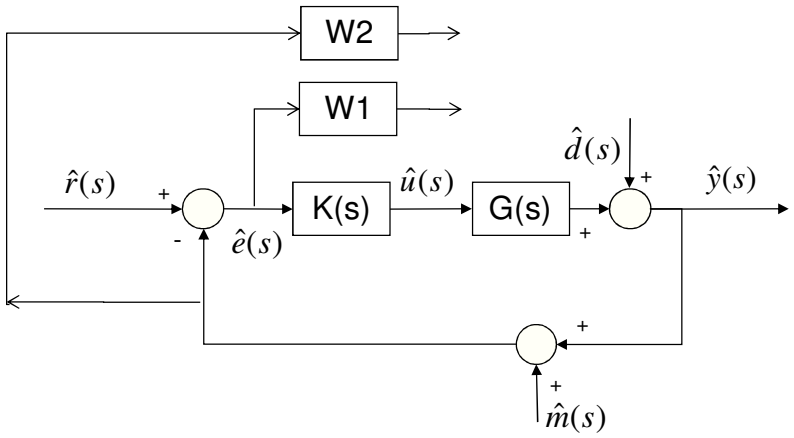

Figure 7: H-Infinity Control Architecture

The H-infinity controller is sized to minimize an objective function as shown in equation 2

$$\min\left\|H(s)\right\|_{\infty} \qquad [2]$$

The H-infinity architecture in Figure 7 is setup to have good reference tracking at some frequencies and good noise rejection at other frequencies. For this architecture, the objective function in equation 2 becomes the following

$$H(s) = \begin{bmatrix} W1(s)S(s) \\ W2(s)T(s) \end{bmatrix} \quad [3]$$

S(s) is the sensitivity function which is a measure of the noise rejection capability. It is defined as follows

$$S(s) = (I + G(s)K(s))^{-1} \quad [4]$$

T(s) is the complementary sensitivity function and it is a measure of the reference tracking capability

$$T(s) = I - S(s) \quad [5]$$

In the objective function in equation 3, both the S(s) and T(s) are weighted by W1 and W2 at different frequencies. Using an optimizer, the controller is then sized to minimize the objective function. The objective function in H-infinity control is not just limited to reference tracking and noise rejection and can include a multitude of other terms (SKOGESTAD and POSTLETHWAITE 2005). If different terms are added or removed from the objective function, the placement of the weighting terms shown in Figure 7 will change.

Since the controller is designed to meet robustness metrics, the H-infinity controller obviously performs quite robustly in the presence of uncertainty. The penalty to obtain such a robust controller is much higher order controller. This results in issues in scheduling the gain of the controller across the flight envelope. Order reduction techniques allow for a simpler controller while maintaining most of the robust properties. H-infinity control has been studied on both turbofan propulsion and STOVL applications (ADIBHATLA, COLLIER and GARG 1998)(GARG, MATTERN, et al. 1990).

2.2.2.3 Linear Quadratic Gaussian (LQG)

As opposed to H-infinity control which is designed to meet certain robustness metrics, a LQG control is sized to provide the most optimal control path in the presence of process and sensor uncertainty. Optimal control is defined as control that provides an

identical response to changes in demand independent of the changes in the dynamics of the system. Figure 8 below shows the LQG control architecture. The use of a state estimator limits LQG control to model based applications.

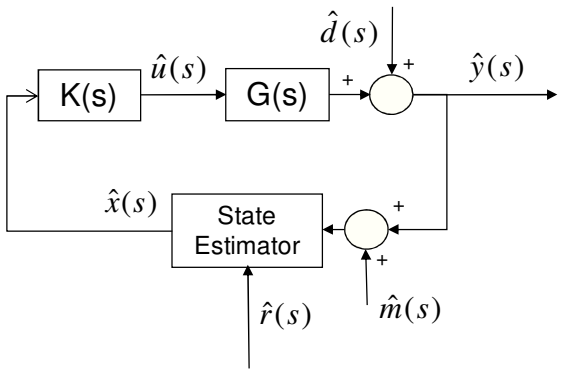

Figure 8: LQG Control Architecture

The controller is sized such that given a state feedback an optimal control movement can be determined as follows:

$$\hat{u}(s) = -K_r \hat{x}(s) \qquad [6]$$

The gain of the control and estimation of the state is calculated by minimizing the probabilistic objective function in equation 7.

$$J = E\left[\lim_{T\to\infty} \frac{1}{T}\int_0^T (x^T Q x + u^T R u)dt\right] \qquad [7]$$

The calculation of the gain and the estimation of the state is broken up into two components using the separation principle (SKOGESTAD and POSTLETHWAITE 2005) . A state estimator is first used to estimate the unmeasured state of the system. The state estimator incorporates both process and sensor uncertainty. Then a deterministic form of equation 7 is solved to find K_r.

Although the controller is sized to provide optimal performance, robust properties are not ensured. To ensure robustness a loop transfer recover (LTR) process is used. As with the H-infinity methodology, using the LQG methodology results in a fairly high order controller and requires order reduction techniques to make it simpler. In the

literature there are multiple studies using LQG on turbofan engines (GARG 1989) . A recent survey paper outlined many of potential different model based methods demonstrated over the years (LITT, et al. 2004). One of the methods, model predictive control (MPC), offers some additional benefits that other model based methods do not.

2.2.2.4 Model Predictive Control

Model Predictive Control (MPC), in comparison to other model based methods, provides the additional benefit of controlling the engine based upon a constrained optimized prediction of the future path. Non-predictive control systems are reactive systems that adjust the actuators based upon the difference between a measured and reference (or target) parameter at a previous time instance. When encountering a constraint, the controls may not adjust until the constraint is hit resulting in an overshoot of the constraint and an increased risk of a potential unsafe condition. This may lead the designer to increase the margin on the constraints to ensure the potential adverse condition may never occur, resulting in a less than optimal solution. Since the MPC algorithm is a constrained optimization, part of the optimized solution is that the constraints are not violated at any future time-step. This factor ensures that the system will operate more safely and potentially at a more optimal solution, thus making MPC a very attractive option. Equation 8 shows the objective function that is minimized to determine the control move. This function is virtually identical to the function minimized in the LQG controller. The main difference is the addition of the constraints of equation 9 directly into the optimization.

$$V(x) = \sum_{i=1}^{H_p} \left\| \hat{z}(k+i/k) - \hat{r}(k+i) \right\|_{Q(i)}^2 + \sum_{i=0}^{H_u-1} \left\| \Delta\hat{u}(k+i/k) \right\|_{R(i)}^2 \quad [8]$$

Subject to

$$const < \hat{z}(t) < const \qquad [9]$$
$$0 < \Delta\hat{u}(t) < const$$
$$const < \hat{u}(t) < const$$

Figure 9 below shows the architecture for MPC. Compared with non-model based or the general model based control methods, the reference signal over a future prediction horizon and the gain is calculated onboard by doing an optimization. Similar to model based controls, there is an engine model onboard which adds to the computational burden. The main drawback of MPC is the significantly larger computational burden. Since MPC does a constrained optimization over a prediction horizon, the burden is orders of magnitude larger as compared with traditional model based control methods. When put in the context of the tip-jet reaction drive system, with a control sampling rate approximately 20ms, this issue presents a significant obstacle. Additionally, a MPC algorithm has never been validated or shown to operate in real-time on an aerospace application.

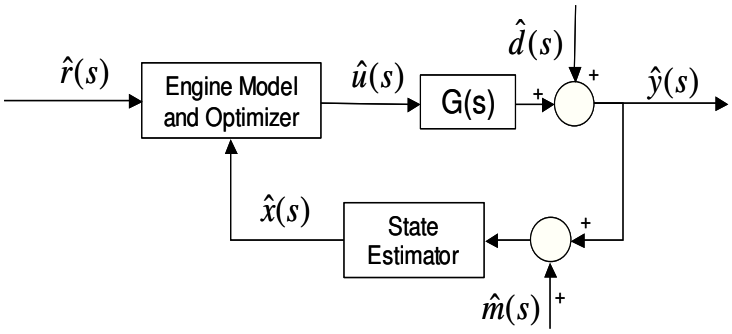

Figure 9: Model Predictive Control (MPC) Architecture

2.3 **Tip-Jet Multivariable Control Feasibility Analysis**

Based upon the descriptions of the different multivariable control methodologies discussed above, a preliminary feasibility study can be performed. Aside from the performance related metrics discussed in Section 1.2, other metrics such as gain scheduling and computational burden need to be accounted for. Additionally costs incurred in making and designing the control as well as maintenance costs are all very

important as well. Lastly the amount of verification and validation already performed on the control is important. This provides an estimate in how ready for show the control technology is. These metrics are not all inclusive but provide a good high level basis to compare the different control methodologies. Table 1 below provides an estimate as to how well each control methodology performs relative to a given metric. These estimates are based upon the authors best understanding of the different methodologies obtained from reviewing the literature.

Table 1: Tip-Jet Reaction Drive Control Feasibility Metrics

Tip-Jet Reaction Drive Control Feasibility Study	Non-Model Based PI	Model Based PI	Non-Model Based H-Infinity	Model Based H-infinity	Linear Quadratic Gaussian	Model Predictive Control
Direct Parameter Control						
Optimal Control						
Performance						
Robustness						
Constraint Handling						
Gain Scheduling						
Computational Burden						
Cost						
Verification and Validation						
Maintenance Costs						

Here it is seen that a non-model based PI controller has the worst performance based metrics but has significant cost benefits. Adding a model to the control improves some of the performance based metrics at the expense of higher costs and computational burden. Using H-infinity adds very good robustness characteristics, but still has high costs involved. LQG adds optimality to the performance, but incurs added costs. MPC is almost the mirror opposite of the non-model based PI controller – great performance and high costs. It has all the good performance metrics of LQG and additionally very good constraint handling. MPC has significant limitations as a result of the large computational burden associated with constrained optimization. If the computational burden can be reduced, the ability to control to a constrained optimized future path makes MPC an attractive alternative.

CHAPTER 3

MODEL PREDICTIVE CONTROL

3.1 **MPC Basics**

As mentioned in the previous chapter, an MPC algorithm performs a constrained optimization over a predicted horizon. The optimization routine attempts to minimize a cost function that may contain many factors. Typical MPC algorithms track a reference or target value over a prediction horizon as seen in Figure 10. For a system such as an aircraft engine, the reference value would be a thrust profile. Additionally, the objective function may minimize the movements of an actuator over a control horizon. This would prevent the actuators from wildly moving or even oscillating from one time-step to the next. Under the assumptions of a fixed reference profile, a perfect system model, no system disturbances, or no output measurement errors, the MPC algorithm can be run once and the predicted control settings input at each appropriate time-step. However, these assumptions are not realistic and require the MPC algorithm to be run at every time-step with only the first control setting input at each time-step. This type of MPC application is commonly referred to as receding horizon control (MACIEJOWSKI 2002), and in this type of control, the prediction horizon steps into the future (i.e., recedes into the horizon) as the MPC algorithm steps in time.

In theory, the prediction horizon of the MPC algorithm can extend to infinity. This would ensure that the control system is stable. However this would result in an unrealistically large computational burden. To minimize this computational burden, the prediction horizon should be at least long enough to capture the response time of the system. The control horizon shown in Figure 10 is the planned control movements over the prediction horizon. The control horizon can be less than or equal to the length of the prediction horizon. However if the control horizon is too short, this may result is a less

optimal solution. The optimization is constrained against various value and rate limitations of both the input and predicted parameters. The first constraint is on the absolute value of a model-calculated output. The other constraints are placed on the rate of change and the absolute value of the control inputs, respectively.

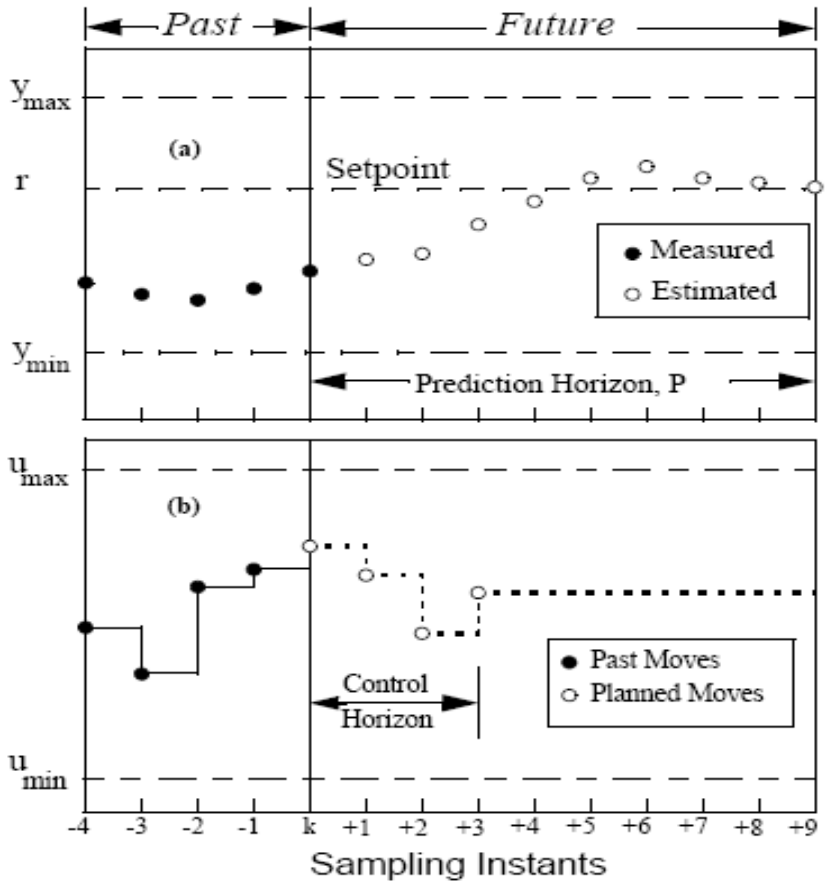

Figure 10: Model Predictive Control (BEMPORAND, MORARI and RICKER 2007)

A typical cost function for an aerospace application that a MPC algorithm minimizes is generally of the following form (MACIEJOWSKI 2002):

$$V(x) = \sum_{i=1}^{H_p} \left\| \hat{z}(k+i/k) - \hat{r}(k+i) \right\|_{Q(i)}^2 + \sum_{i=0}^{H_u-1} \left\| \Delta\hat{u}(k+i/k) \right\|_{R(i)}^2$$

Where

$\hat{z}$ are model calculated parameters
$\hat{r}$ are reference or target values
$\hat{u}$ are actuator positions
H_p is the prediction horizon
H_u is the control horizon
Q and R are weighting matrices

Subject to these constraints

$$const < \hat{z}(t) < const$$
$$0 < \Delta\hat{u}(t) < const$$
$$const < \hat{u}(t) < const$$

Governed by state dynamics model

$$\hat{x}(t+1) = A\hat{x}(t) + B\hat{u}(t) + B_d\hat{v}(t)$$
$$\hat{z}(t+1) = C\hat{x}(t+1) + D_d\hat{v}(t)$$

Governed by control input equations

$$\hat{u}(t) = \hat{u}(t-1) + \Delta\hat{u}(t)$$

3.2 **Literature Review**

Over the last 30 years, MPC has been used for a variety of industrial applications; Qin provides a good overview of some of the applications. In general MPC can be broken down into two different classes of methodologies: linear and non-linear(QIN and BADGWELL 2003). Maciejowski and Camacho provide a good overview of linear theory and methods while Allgower provides an overview of non-linear methods (MACIEJOWSKI 2002)(CAMACHO and BORDONS 2004)(ALLGOWER and ZHENG 2000). Linear methods typically result in less computational burden than non-linear methods; however, a linear approximation may not be applicable for all types of systems. The next few paragraphs provide an extensive, though not completely exhaustive, overview of MPC applications done on systems with similar dynamics to the tip-jet reaction drive system as well as systems that have subsystems with varying levels of interactions.

Over the last 10 years MPC has been a focus of research on the gas turbine, which has system dynamics very similar to the tip-jet reaction drive system. General Electric performed one such study, funded by NASA (VIASSOLO, KUMAR and BRUNELL 2007). Its purpose was to demonstrate an MPC algorithm for fuel consumption minimization across a mission. In this algorithm, a centralized MPC varies the fuel flow and variable stator vanes (VSV) to track thrust demand across a mission. The authors used a linearized model of the engine at the current operating conditions to perform the optimization. Similarly, the United Technology Research Center developed a MPC algorithm concept for gas turbine jet engines (FULLER, SETO and MEISNER 2000). In this algorithm, fuel flow, stator vane angle, and three different compressor bleeds were optimized during an engine transient to track a demanded thrust. A piecewise linear model was used to approximate the non-linear engine performance. The performance of the control was better when compared with a baseline engine control. However, this application did not operate in real time. Recently, one of the authors of the paper registered a patent that claims a 10-30 fold reduction in computational burden relative to current state of the art (J. W. FULLER 2007). Mu and Rees also proposed a MPC for a turbofan that varies fuel flow to track a demanded thrust profile (MU, REES and LIU 2004). In this application, the authors used a neural network to estimate the linearized model of the gas turbine. More recently, Richter proposed a multiplexed MPC algorithm for a turbofan engine (RICHTER, SINGARAJU and LITT 2008). This work was based off the multiplexed MPC work of Ling and Maciejowski (LING, MACIEJOWSKI and WI 2005). The authors varied fuel flow, variable stator vanes, and exhaust nozzle area to track a demanded thrust. In an attempt to reduce the computational burden of the MPC problem, the authors optimized the inputs sequentially (multiplexed) as the engine stepped in time. The authors also used a linearized model of the engine similar to Viassolo to solve the MPC optimization problem. In all these applications, real-time

performance was not demonstrated, though the multiplexed MPC did demonstrate a significant computational burden reduction relative to other methods.

Besides turbofan applications, the gas turbine in general has also been a focus of MPC applications. Van Essen described a MPC algorithm on a lab gas turbine. In this algorithm, the fuel flow of a laboratory gas turbine is scheduled to avoid certain operability constraints, such as engine surge during an engine transient (VAN ESSEN and DE LANGE 2001). In order to alleviate the computational burden, the dynamics of the turbine were linearized at each operating point. Using a control sampling rate of 1.2s, real-time capability was demonstrated. However, a control sampling rate of this magnitude is much too large to be suitable for the tip-jet reaction drive system. A more recent application of MPC on a gas turbine was done by D'Amato whose focus was on using MPC for fast combined cycle power plant startups while avoiding stress constraints on the system components. In this application, gas turbine fuel flow and airflow are optimized during the system startup to minimize engine startup time(D'AMATO 2006). Although the authors note that the gas turbine system is highly non-linear, the computational burden reduction from linearizing the gas turbine was very beneficial and actually an enabling technology. The prediction horizon for this application was approximately 50 minutes with a control sampling rate chosen to realize real-time operation (though it is not clearly stated what that rate is).

Although the gas turbine system is non-linear, all the applications mentioned above used a linear approximation of the turbine to perform the optimization. This assumption allowed the use of computationally much less intensive linear MPC methods. Even with a linear assumption, real-time performance of the MPC algorithm was not demonstrated on the gas turbine applications where the control sampling rate was small (20 ms). However, multiplexed MPC has demonstrated a significant computational burden reduction relative to the other multivariable control applications. Based upon this fact, multiplexing is a very attractive method for application on systems similar to the tip-

jet reaction drive with similar dynamics and more control inputs than a single gas turbine if further improvements can be made. Although many studies have been done to prove the MPC concept on systems with similar dynamics to the tip-jet reaction drive system, but these studies have succeeded in proving that many challenges still remain.

Since the control sampling rates for the turbofan and tip-jet reaction drive system are similar (due to similar dynamics), the MPC for each has execute in a similar amount of time. For a system like the tip-jet reaction drive system which has more control inputs than a turbofan engine, solving the optimization problem of a large centralized MPC algorithm will have a much larger computational burden relative to a single turbofan. Therefore MPC methods which reduce the computational burden are of interest.

Many methods over the years have been developed that attempt to reduce the computational burden of the MPC problem though only a few highlighted methods will be covered in this dissertation. As mentioned above, the multiplexed MPC algorithm developed by Ling and Maciejowski, which updates the control inputs sequentially, has been demonstrated to provide a computational burden reduction relative to a centralized MPC algorithm(LING, MACIEJOWSKI and WI 2005). In Ling and Maciejowski, the authors propose two specific multiplexing methods. The first method breaks up the control inputs into multiplexed subsets in time and optimizes each subset. Although attractive in terms of control performance, this method does not offer significant computational burden reduction. The second method only optimizes a smaller subset of control inputs while holding the non-optimized inputs constant or set to some fixed path. This method potentially offers significant reduction in computational burden, though the performance of the control may not be optimal relative to a centralized MPC algorithm.

Another popular method has been developed by Bemporand (BEMPORAND, MORARI and DUA, et al. 2002). In this method, the MPC optimization is done off-line based upon the state of the system. Lookup tables are used to define the control input path over the prediction horizon. This method offers significant computational burden

reduction; however as the number of states on the system become large, the use of tables can become very cumbersome. Both of these methods offer significant performance reduction and are attractive options for implementing on systems similar to the tip-jet reaction drive.

The next two methods that will be described were developed to provide a quick answer that approaches the equilibrium (or target) and is stable; however, the performance may be less than optimal. Knowing that the system of interest has a small control sampling rate, getting a quick stable answer is potentially an attractive option. The first method is known as a dual-mode controller (SCOKAERT, MAYNE and RAWLINGS 1999). MPC control is used to get the system close to equilibrium and then the controller switches over to a simple linear feedback control. This method is very attractive because it provides a simple way to ensure stability. However, due to use of a linear feedback controller close to equilibrium instead of a MPC algorithm, the performance of the control may not be optimal. The second method focuses on providing a quick answer that ensures stability approaching the equilibrium. It is known as contractive MPC (KOTHARE and MORARI 2000). In this method, a contractive constraint is added to the algorithm that states only that the system must be closer to equilibrium from the prior time-step by a certain constraint factor. As with the dual mode control, this method may provide less than optimal control performance but with the assurance of a quick stable answer.

The tip-jet reaction drive system, which has two engines linked to a rotor, can be viewed as a system with a set of subsystems interacting with each other. For this type of system, a group of MPC methods known as distributed MPC become of interest. In these methods, each subsystem has its own MPC optimization problem. An additional benefit from using a distributed MPC formulation is that each MPC problem may become smaller (in terms of control inputs to optimize) and therefore the computational burden can be reduced. The main issues that arise from use of these methods is the ability to

handle subsystem interactions and how often information between the subsystems is exchanged. These methods have often been applied to control of vehicles moving together in a formation. Keviczky proposed a method where each subsystem MPC algorithm calculates the optimal control path for itself and its adjacent neighbors (KEVICZKY, BORRELLI and BALAS 2004). Stability for this algorithm has been proven using assumptions that are fairly limiting. Depending on the number of neighbors a subsystem has, the dimension of the MPC problem for this algorithm may also become large. Dunbar proposed an algorithm in which each individual subsystem optimizes only its own control inputs(DUNBAR 2007). To handle the interactions between each subsystem, an assumed control input path for all the subsystems is defined. To ensure stability, an additional control capability constraint is added that states that the optimized control path cannot deviate by more than a defined limit from the assumed control input. In this algorithm, the MPC problem is small relative to a centralized MPC problem resulting from optimizing only the subsystem control inputs. However, the control capability constraint may result in suboptimal performance relative to a centralized MPC. Richards proposed a distributed MPC algorithm similar to the multiplexed MPC algorithm, where each subsystem optimization is done sequentially (RICHARDS and HOW 2007)(TRODDEN and RICHARDS 2006). To handle the interactions, each subsystem's inputs are assumed known and can be viewed as a known disturbance. This algorithm focuses on the fact that communication between parallel MPC algorithms may not be feasible. The size of the optimization problem is small relative to the centralized control. Lastly, Venkat proposed a distributed MPC algorithm that optimizes each subsystem individually but in parallel (VENKAT 2006). To handle interactions between subsystems, information is exchanged between each subsystem controller after each iteration. The optimization problem is small relative to a centralized control, however the iteration adds computational burden. Fortunately, feasibility and stability of this algorithm are proven even with one iteration at the expense of control performance.

CHAPTER 4

TECHNICAL CHALLENGES, RESEARCH QUESTIONS, AND HYPOTHESIS

Now that the motivation for the research has been discussed and the theories of model predictive control presented, in this chapter it is necessary to discuss the design constraints involved in determining a feasible design space for a MPC algorithm. In the process of these discussions, many technical challenges will be encountered. The investigation of design constraints along with the technical challenges will lead to the formulation of the first research question. In order to answer this question, a multiplexed MPC algorithm will be explored, thus leading to a second research question. In an attempt to answer the second research question, feasible cooperative MPC will be analyzed. The discussions of the aforementioned MPC algorithms and the combination of their capabilities will lead to the proposing of a hypothesis. In total, the following sections will move from a theoretical and generalized discussion of MPC to a realistic application of a practical technique to the system in question.

4.1 MPC DESIGN CONSTRAINTS

The prediction horizon and sampling time-step of the MPC algorithm are defined by the dynamics of the system being controlled. To accurately capture system response, the prediction horizon must be longer than the time constant of the slowest system dynamic(D'AMATO 2006). If the prediction horizon is too short, the resulting control may become unstable or provide suboptimal performance. The sampling time-step should not be much larger than the fastest dynamic of the system; otherwise the model used for MPC may not be accurate enough. As shown below in Figure 11 the infeasible

areas due to the sampling time-step and prediction horizon constraint are shown as the blue and green areas respectively.

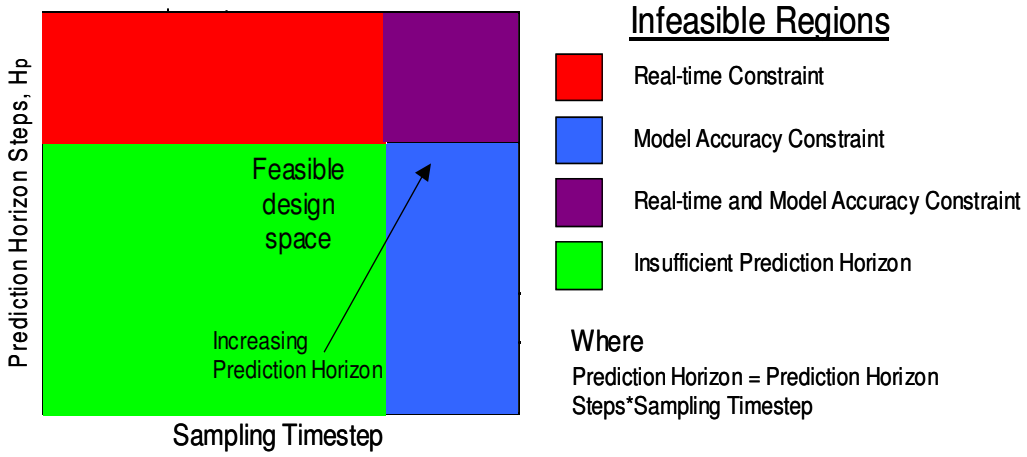

Figure 11: MPC Design Constraints (D'AMATO 2006)

The other constraint on the MPC algorithm is defined by the actual sampling rate of the system control. This constraint defines the minimum amount of time required for the MPC algorithm to execute to determine what the control input for the next time-step should be. To put the problem in perspective, all the necessary calculations and model updates, if done online, need to be done within the time frame of the control-sampling rate. According to Brunell, a typical engine sampling rate is on the order of 20 ms (BRUNELL, BITMEAD and CONNOLLY 2002). From here onward, any reference to "real-time" refers to a time less than or equal to 20 ms. Also anything that will be done online has to have real-time capability (i.e., must execute << 20 ms). An algorithm that executes within this time constraint is considered to be a real-time algorithm. Thusly, the area highlighted in red on Figure 11 is the infeasible area from violating the real-time constraint.

To define a feasible design space, all these design constraints must be taken into account. The system dynamics define feasible regions for the sampling time-step and prediction horizon. It is visibly apparent that as the control sampling rate approaches zero, the real-time constraint moves down on the figure and will eventually result in no

feasible region. However, the designer has some degrees of freedom in defining the real-time constraint.

Traditional controls are very simple and can easily execute within the real-time constraint. However the computational burden for a MPC algorithm is large and there are a number of factors that affect the computational burden and thus the execution time of the algorithm. These factors include the size of the optimization problem, the linearity of the system, the execution time of the model, and the speed of the computer processor. The size of the optimization problem is a function of smaller components: the optimization method, the number of state and control variables, the prediction horizon, and the sampling time-step. The prediction horizon divided by the sampling time-step defines the number of time-steps, H_p, in the algorithm. Interior point methods and active set methods, which are often the most computationally efficient methods, include the state variables in the algorithm, whereas the other method options eliminate the state variables(RAO, WRIGHT and RAWLINGS 1998)(WRIGHT 1997). Although the inclusion of state variables makes the size of the matrices involved in the MPC problem larger, these matrices are sparse and can be decomposed with a potentially significantly reduced computational burden. Additionally the interior point and active set method's computational burden are approximately linear with respect to the prediction horizon, compared with the cubic relationship for the more general methods. An estimate of the computational burden as a function of the number of the control variables (N_u), states (N_x), constraints (N_c), and time-steps (H_p) using the interior point methods is given by Fuller, Kumar, and Miller(FULLER, KUMAR and MILLER, Adaptive Model Based Control of Aircraft Propulsion System 2006). This approximation assumes the prediction and control horizons are equal.

$$CPU \propto H_p * \left[2N_x^3 N_u + 3N_x^2 N_u + \frac{4}{3} N_u^3 + N_c \left(N_x + N_u \right)^2 \right] \qquad [10]$$

Or more generally

$$CPU \propto H_p * \left[N_x + N_u\right]^3 \quad [11]$$

As mentioned previously, the prediction horizon and the sampling time-step are largely a function of the dynamics of the system and are fairly static; thus the number of time-steps, H_p, is defined by the system to be controlled. From equation 11, the computational time is cubic with respect to both the number of state and control parameters to be controlled. It is also linear with respect to the number of prediction time-steps. The tip-jet reaction drive system can be used to help understand the scope of the computational burden. For this system, there are at a minimum six control inputs that can be controlled. Neglecting gas path dynamics there are about eight states. The computational burden for this system would be approximately $2744*H_p$. Depending on the size of the sampling rate of the control, this burden may be quite large, making MPC infeasible. If only one control parameter is controlled, assuming the prediction horizon and the number of states being tracked does not change, the computational burden is approximately an order of magnitude lower. By comparison, controlling three inputs reduces the order by about two. A similar trend is seen with tracking the states. As the number of states being tracked can be reduced, the computational burden further decreases and approaches a two order of magnitude reduction with only one state being tracked. The lower magnitude computational burdens may result in MPC being a feasible alternative. Therefore for tip-jet reaction drive system with a very fast dynamics and small control sampling rate, and in turn a very stringent real-time constraint, the amount of variables that can be controlled in a single MPC algorithm may be limited to a few variables. This leads to the following research question:

How can the dimensions of the MPC problem for the tip-jet reaction drive system be reduced to minimize the computational burden, thus making MPC a more feasible alternative?

4.2 **Centralized MPC**

Before determining how the computational effort of a MPC algorithm can be reduced, a baseline MPC algorithm has to be chosen. A good reference is a centralized control that optimizes all the control inputs at each time over the entire prediction horizon. This type of control has been applied to multiple turbofan applications and stability has been proven. However, real-time capability has not been demonstrated. Additionally, a centralized control, although computationally intensive, has the ability to handle all the subsystem interactions, and therefore, the performance of the control in regards to meeting the objective function, eliminating potential constraint violation, and disturbance rejection should be optimal. These baseline capabilities, whether negative such as computational burden or positive such as control performance, can be used as a reference point for the proposed algorithms. A generalized algorithm for centralized MPC is shown below. The generic objective function Φ is a function of the initial conditions, *u(k-1)* and *x(k)*, and future control moves, *ΔU(k)*. The vector *ΔU(k)* contains all the control inputs over the whole control horizon and has the dimension of H_u*N_u.

Below is an algorithm for centralized MPC.

$$V(k) \overset{\Delta}{=} \min_{\Delta U(k)} \Phi[\Delta U(k), u(k-1), x(k)]$$

Where $$\Delta U(k) = \begin{bmatrix} \Delta\hat{u}(k \mid k) \\ \vdots \\ \Delta\hat{u}(k + H_u - 1 \mid k \end{bmatrix}$$

Subject to these constraints

$$const < \hat{z}(t) < const$$
$$0 < \Delta\hat{u}(t) < const$$
$$const < \hat{u}(t) < const$$

The computational burden of centralized MPC using interior point methods can be approximated as follows:

$$H_p * \left[N_x + N_u\right]^3 \qquad [12]$$

N_u and N_x in this relationship are the total number of control inputs and states for the system. One algorithm mentioned in Chapter 3, multiplexed MPC, is capable of reducing the dimension of N_u and potentially N_x in the general system of interest.

4.3 **Multiplexed MPC**

Multiplexed MPC updates each control input individually or in smaller groups sequentially, thus reducing the number of control input variables at each time-step. The concept behind multiplexed MPC comes from looking at the cubic relationship of the computational burden with the number of control input parameters. If each control input is varied in sequence one by one, the computational burden can be reduced by a factor of up to N_u^3 (assuming $N_x << N_u$). Multiplexed MPC has been demonstrated on a commercial turbofan engine model with three actuators; it was shown to reduce computational burden relative to centralized control by up to a factor of three. The multiplexed control is shown in comparison with traditional MPC in the two graphs in Figure 12. Notice that control update rate is reduced by a factor of N_u. This allows all the control inputs to be updated over the original control sampling rate with the resultant increase in the computational burden proportional to N_u.

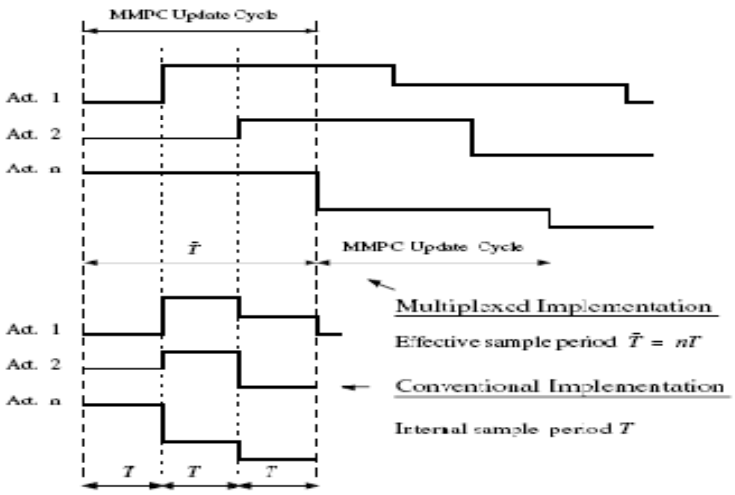

Figure 12: MPC (bottom) Versus Multiplexed MPC (top) (RICHTER, SINGARAJU and LITT 2008)

A generic multiplexed MPC algorithm developed by Ling and Maciejowski is shown below (LING, MACIEJOWSKI and WI 2005). $\Delta U_m(k)$ is a multiplexed subset of $\Delta U(k)$ and has the dimensions H_u (assuming only one control variable is updated at any one time). The control inputs not included in the multiplexed subset are assumed to be constant. It can be seen that when comparing this to the centralized control algorithm that the objective function is identical, but the difference is in the smaller subset of control variables to be optimized. Also since there is a multiplexed subset of control inputs, the constraints on the objective function regarding the control input position and rate change can be reduced to include only the ones that concern themselves with the multiplexed subset.

Below is an algorithm for multiplexed MPC.

$$V(k) \overset{\Delta}{=} \min_{\Delta U_m(k)} \Phi[\Delta U_m(k), u(k-1), x(k)]$$

Where $\Delta U_m(k)$ is a multiplexed subset of $\Delta U(k)$

Subject to these constraints

$$const < \hat{z}(t) < const$$
$$0 < \Delta\hat{u}_m(t) < const$$
$$const < \hat{u}_m(t) < const$$

The following range can approximate the computational burden for multiplexed MPC:

$$H_p * \left[N_x + 1\right]^3 \text{ to } N_u H_p * \left[N_x + 1\right]^3 \quad [13]$$

This reduction in computational burden potentially could be quite significant. However, multiplexed MPC has some limitations. Assuming the same control update rate, the updating of only one control input at a time may result in non-optimal control, and potentially significantly adverse disturbance rejection properties, and would not appear to be a very feasible alternative. However, given the significant reduction in computational burden, the update time of the control could potentially be made smaller (and conversely the prediction horizon larger), since the computational burden is linear with respect to prediction horizon. Reducing the update rate of the control by a factor of N_u will allow each control input parameter to be updated over the original sampling rate but with an increased computational burden proportional to N_u. This large decrease in sampling rate to match centralized control performance may not be feasible because the control physically cannot update quickly enough. Additionally there is not yet a proof for stability on either of these two options. In spite of these limitations, multiplexed MPC is an attractive option for reducing the computational burden of the MPC problem of interest. The application of multiplexing to the system of interest leads to the following research question:

For the tip-jet reaction drive system with highly interactive subsystems and a large number of multiplexed control inputs, how can the MPC problem be broken down or divided to minimize the control performance loss due to multiplexing?

4.4 **Feasible Cooperative MPC**

Feasible cooperative MPC is an attractive option for minimizing the performance loss arising from multiplexing and potentially reducing the computational burden of the system of interest further by understanding how the engines and rotors (subsystems) of the system interact with each other (VENKAT 2006). The tip-jet reaction drive system can be broken up into two different general operation modes of the reaction drive system: diverter valve open and diverter valve closed. Both the dynamics and the interactions of the system are different depending on the mode of operation. When the diverter valve is closed, as shown on the right in Figure 13, it can be plainly seen that the system is broken up into three distinct systems without any interactions. When in this mode, there is no need to use a large computationally intensive centralized MPC. Therefore each of these three systems can be viewed as a separate multivariable control problem. This fact leads to the conclusion that for this mode of operation, three separate MPC instantiations, or more generally a distributed MPC, would be most appropriate. Since there are no interactions, the objective functions of each subsystem (e.g., engine thrust tracking) only contain the value specific to that subsystem. For the tip-jet reaction drive system, if the issue of controlling the diverter valve is ignored, the use of a distributed MPC would create three separate MPC controllers with each controlling two control inputs apiece (the fuel and nozzle actuators of each system). In addition, the number of states that need to be tracked for each of the MPC controllers can be reduced too. Comparing this configuration to a large centralized control, the computational time is reduced by more than one order of magnitude.

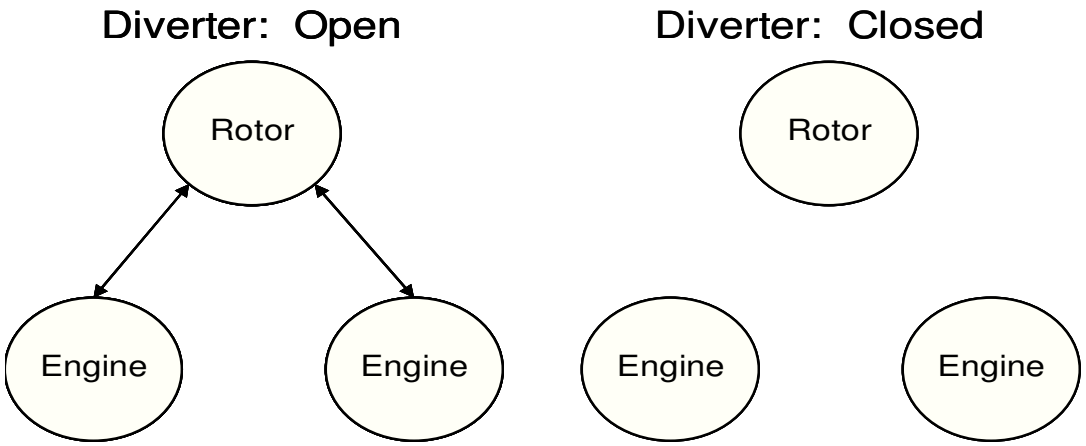

Figure 13: Tip-jet Reaction Drive System Modes of Operation

Below is an algorithm for distributed MPC.

$$V_j(k) \overset{\Delta}{=} \min_{\Delta U_j(k)} \Phi[\Delta U_j(k), u_j(k-1), x_j(k)]$$

Where $\Delta U_j(k)$ are the control inputs for subsystem, j

Subject to these constraints

$$const < \hat{z}_j(t) < const$$
$$0 < \Delta\hat{u}_j(t) < const$$
$$const < \hat{u}_j(t) < const$$

The following is the approximate computational burden for distributed MPC with no interactions:

$$H_p * \left[N_{x,j} + N_{u,j}\right]^3 \qquad [14]$$

Where $N_{u,j} < N_u$ and $N_{x,j} < N_x$

For the diverter closed mode, a simple distributed MPC algorithm shown above would provide optimal and stable control of the tip-jet reaction drive system with over an order of magnitude reduction in the computational burden. However, when the system is in diverter open mode, the interactions between the systems may cause the distributed MPC algorithm to become instable and potentially infeasible. Venkat proposed a distributed control methodology called feasible cooperative MPC that provides both a

stable and feasible distributed MPC control (VENKAT 2006). This method builds on the concept of what Venkat describes as a shared or "cooperative" MPC algorithm where each subsystem's objective function is the sum of all the other subsystem's objective function, while only optimizes the subsystem of interests. The control inputs from the other subsystems previous optimization are then integrated as a known disturbance. To overcome the stability and infeasible issues with cooperative MPC, Venkat proposed adding iterations, p, to the cooperative algorithm at each time-step. The optimization performed each iteration is identical to the cooperative MPC algorithm, but the information from the other control inputs is updated after each iteration. As the number of iterations approach infinity, the solution of this scheme approaches the optimal solution contained in the large centralized algorithm. Venkat proved that this algorithm was stable and provided a feasible answer no matter how times the solution iterates.

Below is an algorithm for feasible cooperative MPC.

$$V_j(k) \overset{\Delta}{=} \min_{\Delta U_j(k)} \sum_{r=1}^{N_d} w_r \Phi_r [\Delta U_1^{p-1}(k), ..., \Delta U_j^{p}(k), ..., \Delta U_{N_d}^{p-1}(k), u(k-1), x(k)]$$

Where $\Delta U_j(k)$ are the control inputs for subsystem, j

Subject to these constraints

$$const < \hat{z}(t) < const$$
$$0 < \Delta \hat{u}_j(t) < const$$
$$const < \hat{u}_j(t) < const$$

Since all the subsystem objective functions are included in the feasible cooperative MPC algorithm, the number of states that need to be tracked in the algorithm may increase. If the interactions between the subsystems are fairly large, the number of states that need to be tracked may approach the overall number of states in the system, N_x.

The algorithm performs an identical optimization each iteration resulting in a computational burden increase proportional to the number of iterations. The following is the approximate computational burden range for feasible cooperative MPC:

$$pH_p * \left[N_{x,j} + N_{u,j}\right]^3 \text{ to } pH_p \left[N_x + N_{u,j}\right]^3 \quad [15]$$

Where $N_{u,j} < N_u$ and $N_{x,j} < N_x$

Depending on the interactions of the subsystems, this reduction in computational burden potentially could be quite significant and makes feasible cooperative MPC an attractive option. However, if the interactions are very large and the number of iterations increases, the associated savings in computational time relative to the centralized MPC start to decrease and potentially become non-existent.

4.5 **Feasible Cooperative Multiplexed MPC**

Both the above-mentioned algorithms provide potential for reducing the computational burden of the MPC problem for the system of interest. When implemented in a centralized format, multiplexed MPC, although offering potentially significant computational burden reduction, has limitations regarding disturbance rejection and physical control issues with the reduction in sampling rate. When using multiplexed MPC in conjunction with feasible cooperative MPC, those limitations can be alleviated to a certain degree. Each subsystem's optimization is done with a smaller set of control inputs. If this smaller subset is multiplexed, the control sampling rate would be reduced by a factor proportional to $N_{u,j}$ as opposed to N_u. This may be a more feasible result. If the control sampling rate is not changed, the use of multiplexed MPC may not have adverse disturbance rejection properties. In addition to the alleviating some of the centralized multiplexed MPC issues, the use of multiplexing with a feasible cooperative MPC algorithm may further reduce the computational burden of the MPC problem. All

the computational burden reduction aspects of both methods can be incorporated into the combined algorithm.

Below is an algorithm for feasible cooperative multiplexed MPC.

$$V_j(k) \overset{\Delta}{=} \min_{\Delta U_{m,j}(k)} \sum_{r=1}^{N_d} w_r \Phi_r [\Delta U_{m,1}^{p-1}(k),...,\Delta U_{m,j}^{p}(k),...,\Delta U_{m,N_d}^{p-1}(k), u(k-1), x(k)]$$

Where $\Delta U_{m,j}(k)$ are the multiplexed subset of control inputs for subsystem, j

Subject to these constraints

$$const < \hat{z}(t) < const$$
$$0 < \Delta\hat{u}_{m,j}(t) < const$$
$$const < \hat{u}_{m,j}(t) < const$$

The following is the approximate computational burden for feasible cooperative multiplexed MPC:

$$pH_p * [N_{x,j} + 1]^3 \text{ to } N_{u,j} pH_p * [N_x + 1]^3 \qquad [16]$$

Where $N_{u,j} < N_u$ and $N_{x,j} < N_x$

Based on the above descriptions, a hypothesis is proposed that attempts to answer the two research questions posed above regarding computational burden reduction and control performance.

For the tip-jet reaction drive system, combining multiplexing and feasible cooperative control into a single algorithm will reduce the computational burden of the MPC problem from approximately $(N_u + N_x)^3$ to $p(N_x+1)^3$ while maintaining similar control performance to a centralized MPC algorithm.

The next chapter of this proposal will focus on a detailed description of the proposed method and how the matrices associated with solving the MPC problem are affected by incorporating the proposed method. From that analysis, conclusions can be drawn to computational reduction obtained by incorporating the proposed controller.

CHAPTER 5

COMPUTATIONAL BURDEN

The approach hypothesized in the last chapter stated that the order of magnitude on the computational burden can be reduced from $(N_u + N_x)^3$ to $p(N_x+1)^3$. To gain a better understanding of the relationship between the computational burden and the input variables, the MPC problem can be examined in terms of the matrices involved. Similarly to the last chapter, the centralized MPC problem will be used as a baseline and the multiplexed, feasible cooperative, and feasible cooperative multiplexed MPC methods will be compared to that baseline.

5.1 **Centralized MPC**

The generic objective function for centralized MPC is shown below. The objective is minimized by adjusting the future state and control inputs.

$$V(k) \overset{\Delta}{=} \min_{\Delta U(k),x(k)} \Phi[\Delta U(k), x(k), u(k-1), x(k)] \qquad [17]$$

For aerospace applications, the objective function is often made up of two terms. The former is the reference tracking term over the prediction horizon. The latter term attempts to minimize the control movement terms over the control horizon.

$$V(k) \overset{\Delta}{=} \sum_{i=1}^{H_p} \left\| \hat{z}(k+i/k) - r(k+i) \right\|_{Q(i)}^2 + \sum_{i=1}^{H_u} \left\| \Delta\hat{u}(k+i-1/k) \right\|_{R(i)}^2 \qquad [18]$$

The following approximate linear relationship between the state and output.

$$\hat{z}(k+i \mid k) = C\hat{x}(k+i \mid k) \qquad \text{for} \qquad i = 1...H_p \qquad [19]$$

Substituting equation 19 into the equation 18, the objective function then becomes as follows.

$$V(k) = \sum_{i=1}^{H_p} \left\| C\hat{x}(k+i/k) - r(k+i) \right\|_{Q(i)}^2 + \sum_{i=1}^{H_u} \left\| \Delta\hat{u}(k+i-1/k) \right\|_{R(i)}^2 \qquad [20]$$

Expanding the norms terms in the above equation

$$V(k)=\sum_{i=1}^{H_p}\begin{bmatrix}(C\hat{x}(k+i\,|\,k))^T Q(i)(C\hat{x}(k+i\,|\,k))\\ +(C\hat{x}(k+i\,|\,k))^T Q(i)r(k+i)\\ +(r(k+i))^T Q(i)(C\hat{x}(k+i\,|\,k))\\ +(r(k+i))^T Q(i)r(k+i)\end{bmatrix}+\sum_{i=1}^{H_u}\Delta\hat{u}(k+i-1/k)^T R(i)\Delta\hat{u}(k+i-1/k) \qquad [21]$$

The second through fourth terms can be viewed as a driving function and can be simplified as a function of the state terms

$$\sum_{i=1}^{H_p}\begin{bmatrix}(C\hat{x}(k+i\,|\,k))^T Q(i)r(k+i)\\ +(r(k+i))^T Q(i)(C\hat{x}(k+i\,|\,k))\\ +(r(k+i))^T Q(i)r(k+i)\end{bmatrix}=\sum_{i=1}^{H_p} f(i)\hat{x}(k+i\,|\,k) \qquad [22]$$

This simplifies the objective function

$$V(k)=\sum_{i=1}^{H_p}\begin{bmatrix}(C\hat{x}(k+i\,|\,k))^T Q(i)(C\hat{x}(k+i\,|\,k))\\ +f(i)\hat{x}(k+i\,|\,k)\end{bmatrix}+\sum_{i=1}^{H_u}\Delta\hat{u}(k+i-1/k)^T R(i)\Delta\hat{u}(k+i-1/k) \qquad [23]$$

There are inequality constraints on the rate of control input change, the actual control input position, and the output of the system

$$E\Delta\hat{u}(k+i-1\,|\,k)\le Y_1 \qquad \text{for} \qquad i=1...H_u \qquad [24]$$

$$F\hat{u}(k+i-1\,|\,k)\le Y_2 \qquad \text{for} \qquad i=1...H_u \qquad [25]$$

$$\hat{z}_c(k+i\,|\,k)=C_c\hat{x}(k+i\,|\,k)\le Y_3 \qquad \text{for} \qquad i=1...H_p \qquad [26]$$

The system dynamics are included in the MPC algorithm as equality constraints.

$$\hat{x}(k+i\,|\,k)=A\hat{x}(k+i-1\,|\,k)+B\hat{u}(k+i-1\,|\,k) \qquad \text{for} \qquad i=1...H_p \qquad [27]$$

$$\hat{u}(k+i-1\,|\,k)=\hat{u}(k+i-2\,|\,k)+\Delta\hat{u}(k+i-1\,|\,k) \qquad \text{for} \qquad i=1...H_u \qquad [28]$$

Grouping the objective function and constraint terms together results, the constrained centralized MPC algorithm can be written as follows.

$$V(k)\overset{\Delta}{=}\min_{\Delta U(k)}\sum_{i=1}^{H_p}\begin{bmatrix}(C\hat{x}(k+i\,|\,k))^T Q(i)(C\hat{x}(k+i\,|\,k))\\ +f(i)\hat{x}(k+i\,|\,k)\end{bmatrix}+\sum_{i=1}^{H_u}\Delta\hat{u}(k+i-1/k)^T R(i)\Delta\hat{u}(k+i-1/k)$$

Subject to these constraints

$$
\begin{aligned}
&E\Delta\hat{u}(k+i-1\,|\,k) \le Y_1 && && i=1...H_u\\
&F\hat{u}(k+i-1\,|\,k) \le Y_2 && && i=1...H_u\\
&C_c\hat{x}(k+i\,|\,k) \le Y_3 && \text{for} && i=1...H_p\\
&\hat{x}(k+i\,|\,k) = A\hat{x}(k+i-1\,|\,k) + B\hat{u}(k+i-1\,|\,k) && && i=1...H_p\\
&\hat{u}(k+i-1\,|\,k) = \hat{u}(k+i-2\,|\,k) + \Delta\hat{u}(k+i-1\,|\,k) && && i=1...H_u
\end{aligned}
$$

To solver the constrained optimization problem, both the equality and inequality constraints are adjoined to the objective function using Lagrange multipliers p(i), q(i), π(i), λ(i), and σ(i) as shown in equation 20. The Lagrange terms for the equality constraints, p(i) and q(i), go to zero at the optimal solution. The inequality constraints are set up such that either the Lagrange term, π(i), λ(i), and σ(i) , is zero or the constraint is active. Note the Lagrange multipliers are now variables used to optimize the objective function. For the sake of brevity, these terms along with the state vector will not be included in the objective function notation. But it can be clearly be seen that as the number of constraints increase, the computational burden increases and this effect will be covered in the summary of each subsequent section.

$$
\begin{aligned}
V(k) \overset{\Delta}{=} & \min_{\Delta U(k),x(k)} \sum_{i=1}^{H_p} \left[\begin{array}{l} \left(C\hat{x}(k+i\,|\,k)\right)^T Q(i)\left(C\hat{x}(k+i\,|\,k)\right) \\ + f(i)\hat{x}(k+i\,|\,k) \end{array} \right] + \\
& \sum_{i=1}^{H_u} \Delta\hat{u}(k+i-1/k)^T R(i)\Delta\hat{u}(k+i-1/k) + \\
& \sum_{i=1}^{H_p} p(i)^T \left[A\hat{x}(k+i-1\,|\,k) + B\hat{u}(k+i-1\,|\,k) - \hat{x}(k+i\,|\,k)\right] + \\
& \sum_{i=1}^{H_u} q(i)^T \left[\hat{u}(k+i-2\,|\,k) + \Delta\hat{u}(k+i-1\,|\,k) - \hat{u}(k+i-1\,|\,k)\right] + \\
& \sum_{i=1}^{H_u} \pi(i)^T \left[E(\Delta\hat{u}(k+i-1\,|\,k) - Y_1\right] + \sum_{i=1}^{H_u} \lambda(i)^T \left[F(\hat{u}(k+i-1\,|\,k) - Y_2\right] + \\
& \sum_{i=1}^{H_p} \sigma(i)^T \left[C_c\hat{x}(k+i\,|\,k) - Y_3\right]
\end{aligned}
\qquad [29]
$$

The optimal solution is when the derivative of the objective function with respect to each input term is zero. Recall when using interior point methods, both the state and control vectors are variables used in the optimization (as opposed to substituting the state

equations out), therefore the partial derivatives with respect to each term need to be taken and set to zero.

$$\frac{\partial V(k)}{\partial \hat{x}(k+i\mid k)} = C^T Q(i)\left(C\hat{x}(k+i\mid k)\right) + f(i) - Ip(i) + A^T p(i+1) + C_c{}^T \sigma(i) = 0 \quad [30]$$

$$\frac{\partial V(k)}{\partial \Delta\hat{u}(k+i-1\mid k)} = R(i)\Delta\hat{u}(k+i-1/k) + Iq(i) + E^T \pi(i) = 0 \quad [31]$$

$$\frac{\partial V(k)}{\partial \hat{u}(k+i-1)} = B^T p(i) + Iq(i+1) - Iq(i) + F^T \lambda(i) = 0 \quad [32]$$

At the optimal solution, the system dynamics equations, which are the equality constraints, have to be satisfied.

$$A\hat{x}(k+i-1\mid k) + B\hat{u}(k+i-1\mid k) - \hat{x}(k+i\mid k) = 0 \quad [33]$$

$$\hat{u}(k+i-2\mid k) + \Delta\hat{u}(k+i-1\mid k) - \hat{u}(k+i-1\mid k) = 0 \quad [34]$$

The inequality constraints can be rewritten into equality constraints by including a slack variable (t_1(i), t_2(i), and t_3(i)) that is zero when the constraint is active and greater than zero when the constraint is not active.

$$E\Delta\hat{u}(k+i-1\mid k) + t_1(i) - Y_1 = 0 \quad [35]$$

$$F\hat{u}(k+i-1\mid k) + t_2(i) - Y_2 = 0 \quad [36]$$

$$C_c\hat{x}(k+i\mid k) + t_3(i) - Y_3 = 0 \quad [37]$$

When the constraint is active, the slack variable is zero, and when the constraint is not active, the Lagrange multiplier is zero. Therefore, the product of the slack variable and the associated Lagrange multiplier has to be zero at all times.

$$T_1\pi + \Pi t_1 = 0 \quad [38]$$

$$T_2\lambda + \Lambda t_2 = 0 \quad [39]$$

$$T_3\sigma + \Sigma t_3 = 0 \quad [40]$$

Where

$$T_1 = diag(t_1(1)\cdots t_1(H_p)) \quad [41]$$

$$T_2 = diag(t_2(1)\cdots t_2(H_p)) \quad [42]$$

$$T_3 = diag(t_3(1)\cdots t_3(H_p)) \quad [43]$$

$$\Pi = diag(\pi(1)\cdots \pi(H_p)) \quad [44]$$

$$\Lambda = diag(\lambda(1)\cdots \lambda(H_p)) \quad [45]$$

$$\Sigma = diag(\sigma(1)\cdots \sigma(H_p)) \quad [46]$$

$$t_1 = \begin{bmatrix} t_1(1) \\ \vdots \\ t_1(H_p) \end{bmatrix} \quad [47]$$

$$t_2 = \begin{bmatrix} t_2(1) \\ \vdots \\ t_2(H_p) \end{bmatrix} \quad [48]$$

$$t_3 = \begin{bmatrix} t_3(i) \\ \vdots \\ t_3(H_p) \end{bmatrix} \quad [49]$$

$$\pi = \begin{bmatrix} \pi(1) \\ \vdots \\ \pi(H_p) \end{bmatrix} \quad [50]$$

$$\lambda = \begin{bmatrix} \lambda(1) \\ \vdots \\ \lambda(H_p) \end{bmatrix} \quad [51]$$

$$\sigma = \begin{bmatrix} \sigma(1) \\ \vdots \\ \sigma(H_p) \end{bmatrix} \quad [52]$$

The product of the slack variable and Lagrange multipliers can be substituted into the inequality constraint equations.

$$E\Delta\hat{u}(k+i-1|k) - \Pi^{-1}(i)T_1(i)\pi(i) - Y_1 = 0 \quad [53]$$

$$F\hat{u}(k+i-1\,|\,k)-\Lambda^{-1}(i)T_2(i)\lambda(i)-Y_2=0 \quad [54]$$

$$C_c\hat{x}(k+i\,|\,k)+\Sigma^{-1}(i)T_3(i)\sigma(i)-Y_3=0 \quad [55]$$

At each time-step, the three partial derivative equations, the two state dynamics equations, and the three inequality constraint equations need to be satisfied. These equations can be written in the matrix form HΘ = R, which can be solved using previously mentioned interior point methods (RAO, WRIGHT and RAWLINGS 1998). The Θ vector contains all the variable used to optimize the objective function and includes control inputs, state variables, and Lagrange multipliers. The H matrix contains all the coefficients associated with the terms of the Θ vector. The R vector contains all the constants. For the purpose of brevity only a one time-step sample of the matrices and vectors is shown, but all other time-steps these matrices and vectors will look identical (except at the first and last time-step which would have truncated versions).

$$\Theta = \begin{bmatrix} \vdots \\ \hat{u}(k+i-2\,|\,k) \\ \Delta\hat{u}(k+i-2\,|\,k) \\ \hat{x}(k+i-1\,|\,k) \\ p(i) \\ q(i) \\ \pi(i) \\ \lambda(i) \\ \sigma(i) \\ \hat{u}(k+i-1\,|\,k) \\ \Delta\hat{u}(k+i-1\,|\,k) \\ \hat{x}(k+i\,|\,k) \\ p(i+1) \\ q(i+1) \\ \vdots \end{bmatrix} \quad [56]$$

$$H=\begin{bmatrix}
\ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & I & E^T & 0 & 0 & 0 & R & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & B^T & -I & 0 & F^T & 0 & 0 & 0 & 0 & 0 & I & 0 \\
0 & 0 & 0 & 0 & -I & 0 & 0 & 0 & C_C^T & 0 & 0 & C^TQC & A^T & 0 & 0 \\
0 & 0 & 0 & A & 0 & 0 & 0 & 0 & 0 & B & 0 & -I & 0 & 0 & 0 \\
0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -I & I & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\Pi^{-1}T_1 & 0 & 0 & 0 & E & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Lambda^{-1}T_2 & 0 & F & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Sigma^{-1}T_3 & 0 & 0 & C_c & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots
\end{bmatrix} \quad [57]$$

$$R=\begin{bmatrix}
\vdots \\
0 \\
0 \\
-f(i) \\
0 \\
0 \\
Y_1 \\
Y_2 \\
Y_3 \\
\vdots
\end{bmatrix} \quad [58]$$

Note all the terms in the H matrix are banded around the diagonal. When the MPC equations are set up in this format, the computational burden of the MPC problem is proportional to the cube of the dimension of the banded portion of the H matrix or the dimension of the Θ vector at a single time-step and linear with the prediction horizon. Table 2 below summarizes the total dimension of the bandwidth of centralized MPC.

Table 2: Centralized MPC Dimensions

Vector	Dimension
$\hat{x}(k+i-1\mid k)$	N_x
$p(i)$	N_x
$q(i)$	N_u
$\pi(i)$	$N_{\Delta\hat{u}c}*N_u$
$\lambda(i)$	$N_{\hat{u}c}*N_u$
$\sigma(i)$	N_{zc}
$\hat{u}(k+i-1\mid k)$	N_u
$\Delta\hat{u}(k+i-1\mid k)$	N_u
Bandwidth Dimension	$(3 + N_{\Delta\hat{u}c}+ N_{\hat{u}c})N_u + 2N_x + N_{zc}$

5.2 **Multiplexed MPC**

For multiplexed MPC, a similar derivation can be made. However, at each time-step, only a smaller subset of inputs is being used, so that the state dynamic equations can be rewritten as follows.

$$\hat{x}(k+i\,|\,k) = A\hat{x}(k+i-1\,|\,k) + B_m\hat{u}_m(k+i-1\,|\,k) \qquad \text{for} \qquad i = 1...H_p \qquad [59]$$

$$\hat{u}_m(k+i-1\,|\,k) = \hat{u}_m(k+i-2\,|\,k) + \Delta\hat{u}_m(k+i-1\,|\,k) \text{ for} \qquad i = 1...H_u \qquad [60]$$

Where $\eta(k) = k \bmod(N_u) + 1$

$\Delta\hat{u}_m(k+i-1|k)$ only changes the $\eta(k)^{th}$ value of the $\Delta\hat{u}(k+i-1|k)$ vector
$\hat{u}_m(k+i-1|k)$ only changes the $\eta(k)^{th}$ value of the $\hat{u}(k+i-1|k)$ vector
B_m is the $\eta(k)^{th}$ column vector of B

Thusly the cost function for a constrained multiplexed MPC can be written.

$$V(k) \overset{\Delta}{=} \min_{\Delta U_m(k)} \sum_{i=1}^{H_p} \begin{bmatrix} \left(C\hat{x}(k+i\,|\,k)\right)^T Q(i)\left(C\hat{x}(k+i\,|\,k)\right) \\ + f(i)\hat{x}(k+i\,|\,k) \end{bmatrix} + \sum_{i=1}^{H_u} \Delta\hat{u}_m(k+i-1/k)^T R(i)\Delta\hat{u}_m(k+i-1/k)$$

Subject to these constraints

$$E_m\Delta\hat{u}_m(k+i-1\,|\,k) \le Y_{1m}$$
$$F_m\hat{u}_m(k+i-1\,|\,k) \le Y_{2m}$$
$$C_c\hat{x}(k+i\,|\,k) \le Y_3$$
$$\hat{x}(k+i\,|\,k) = A\hat{x}(k+i-1\,|\,k) + B_m\hat{u}_m(k+i-1\,|\,k)$$
$$\hat{u}_m(k+i-1\,|\,k) = \hat{u}_m(k+i-2\,|\,k) + \Delta\hat{u}_m(k+i-1\,|\,k)$$

The optimal equations used to solve the MPC problem can then be rewritten in multiplexed format.

$$\frac{\partial V(k)}{\partial\hat{x}(k+i\,|\,k)} = C^T Q(i)\left(C\hat{x}(k+i\,|\,k)\right) + f(i) - Ip(i) + A^T p(i+1) + C_c{}^T\sigma(i) = 0 \qquad [61]$$

$$\frac{\partial V(k)}{\partial\Delta\hat{u}_m(k+i-1\,|\,k)} = R_m(i)\Delta\hat{u}_m(k+i-1/k) + I_m q_m(i) + E_m{}^T\pi_m(i) = 0 \qquad [62]$$

$$\frac{\partial V(k)}{\partial \hat{u}_m(k+i-1)} = B_m{}^T p(i) + I_m q_m(i+1) - I_m q_m(i) + F_m{}^T \lambda_m(i) = 0 \quad [63]$$

$$A\hat{x}(k+i-1\,|\,k) + B_m \hat{u}_m(k+i-1\,|\,k) - \hat{x}(k+i\,|\,k) = 0 \quad [64]$$

$$\hat{u}_m(k+i-2\,|\,k) + \Delta\hat{u}_m(k+i-1\,|\,k) - \hat{u}_m(k+i-1\,|\,k) = 0 \quad [65]$$

$$E_m \Delta\hat{u}_m(k+i-1\,|\,k) - \Pi^{-1}{}_m(i) T_{1m}(i) \pi_m(i) - Y_{1m} = 0 \quad [66]$$

$$F_m \hat{u}_m(k+i-1\,|\,k) - \Lambda^{-1}{}_m(i) T_{2m}(i) \lambda_m(i) - Y_{2m} = 0 \quad [67]$$

$$C_c \hat{x}(k+i\,|\,k) + \Sigma^{-1}(i) T_3(i) \sigma(i) - Y_3 = 0 \quad [68]$$

Similarly to centralized MPC, the equations be written in matrix form of $H_m\Theta_m = R_m$.

$$\Theta_m = \begin{bmatrix} \vdots \\ \hat{u}_m(k+i-2\,|\,k) \\ \Delta\hat{u}_m(k+i-2\,|\,k) \\ \hat{x}(k+i-1\,|\,k) \\ p(i) \\ q_m(i) \\ \pi_m(i) \\ \lambda_m(i) \\ \sigma(i) \\ \hat{u}_m(k+i-1\,|\,k) \\ \Delta\hat{u}_m(k+i-1\,|\,k) \\ \hat{x}(k+i\,|\,k) \\ p(i+1) \\ q_m(i+1) \\ \vdots \end{bmatrix} \quad [69]$$

$$H_m = \begin{bmatrix}
\ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & I_m & E_m^T & 0 & 0 & 0 & R_m & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & B_m^T & -I_m & 0 & F_m^T & 0 & 0 & 0 & 0 & 0 & I_m & 0 \\
0 & 0 & 0 & 0 & -I & 0 & 0 & 0 & C_C^T & 0 & 0 & C^T Q C & A^T & 0 & 0 \\
0 & 0 & 0 & A & 0 & 0 & 0 & 0 & 0 & B_m & 0 & -I & 0 & 0 & 0 \\
0 & I_m & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -I_m & I_m & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\Pi_m^{-1} T_{1m} & 0 & 0 & 0 & E_m & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Lambda_m^{-1} T_{2m} & 0 & F_m & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Sigma^{-1} T_3 & 0 & 0 & C_c & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots
\end{bmatrix} \quad [70]$$

$$R_m = \begin{bmatrix}
\vdots \\
0 \\
0 \\
-f(i) \\
0 \\
0 \\
Y_{1m} \\
Y_{2m} \\
Y_3 \\
\vdots
\end{bmatrix} \quad [71]$$

Table 3 below summarizes the bandwidth dimension for multiplexed MPC. It can be seen that the dimension on the q(i) term, which is the Lagrange multiplier adjoined to the control input dynamic equation, is reduced to one. Also, since the number of control inputs is reduced to one, the dimension of the Lagrange multiplier associated with the inequality constraint terms are reduced also to reflect the smaller number of control inputs. The bandwidth, and in turn the computational burden, for multiplexed MPC is significantly lower than for centralized MPC. However, as the number of control inputs becomes large, the prediction horizon may have to have smaller time-steps to ensure the control performance and disturbance rejection properties are suitable, thus increasing the computational burden. Additionally, if the system is not very interactive, a centralized multiplexed MPC is not an attractive option because the lack of dependence between control inputs and system level outputs.

Table 3: Multiplexed MPC Dimension

Vector	Dimension
$\hat{x}(k+i-1\mid k)$	N_x
$p(i)$	N_x
$q(i)$	1
$\pi(i)$	$N_{\Delta\hat{u}c}$
$\lambda(i)$	$N_{\hat{u}c}$
$\sigma(i)$	N_{zc}
$\hat{u}(k+i-1\mid k)$	1
$\Delta\hat{u}(k+i-1\mid k)$	1
Bandwidth Dimension	$3 + N_{\Delta\hat{u}c} + N_{\hat{u}c} + 2N_x + N_{zc}$

5.3 **Feasible Cooperative MPC**

A similar derivation can be performed with feasible cooperative MPC. The control inputs of the system are divided into smaller subsets associated with each subsystem of the system as shown below.

$$\hat{u}(k+i-1\mid k)=\begin{bmatrix}\hat{u}_1(k+i-1\mid k)\\ \vdots \\ \hat{u}_{N_d}(k+i-1\mid k)\end{bmatrix} \qquad \text{for} \qquad i=1...H_p \qquad [72]$$

$$\Delta\hat{u}(k+i-1\mid k)=\begin{bmatrix}\Delta\hat{u}_1(k+i-1\mid k)\\ \vdots \\ \Delta\hat{u}_{N_d}(k+i-1\mid k)\end{bmatrix} \qquad \text{for} \qquad i=1...H_p \qquad [73]$$

The control input portion of the state dynamics equations can be defined as the summation of inputs from all the subsystems.

$$B\hat{u}(k+i-1\mid k)=\sum_{d=1}^{N_d}B_d\hat{u}_d(k+i-1\mid k) \qquad [74]$$

Where B_d is the smaller subset of B associated with subsystem d.

The state dynamics equations can be rewritten in terms of the control inputs being optimized and the control inputs of the other subsystems not being optimized.

$$\hat{x}(k+i\mid k)=A\hat{x}(k+i-1\mid k)+B_j\hat{u}_j(k+i-1\mid k)+\sum_{d=1,d\neq j}^{N_d}B_d\hat{u}_d(k+i-1\mid k) \qquad \text{for } i=1..H_p \qquad [75]$$

$$\hat{u}_j(k+i-1\,|\,k) = \hat{u}_j(k+i-2\,|\,k) + \Delta\hat{u}_j(k+i-1\,|\,k) \qquad \text{for } i = 1..H_u \qquad [76]$$

The cost function for a constrained feasible cooperative MPC is then written as follows:

$$V_j(k) \overset{\Delta}{=} \min_{\Delta U_j(k)} \sum_{i=1}^{H_p} \left[\begin{matrix} \left(C\hat{x}(k+i\,|\,k)\right)^T wQ(i)\left(C\hat{x}(k+i\,|\,k)\right) \\ + f(i)\hat{x}(k+i\,|\,k) \end{matrix} \right]$$

$$+ \sum_{i=1}^{H_u} \Delta\hat{u}_j^{\,p}(k+i-1/k)^T R_j(i) \Delta\hat{u}_j^{\,p}(k+i-1/k)$$

Subject to these constraints

$$E_j \Delta\hat{u}_j^{\,p}(k+i-1\,|\,k) \le Y_{1j}$$

$$F_j \hat{u}_j^{\,p}(k+i-1\,|\,k) \le Y_{2j}$$

$$C_c \hat{x}(k+i\,|\,k) \le Y_3$$

$$\hat{x}(k+i\,|\,k) = A\hat{x}(k+i-1\,|\,k) + B_j \hat{u}_j^{\,p}(k+i-1\,|\,k) + \sum_{d=1,d\neq j}^{N_d} B_d \hat{u}_d^{\,p-1}(k+i-1\,|\,k)$$

$$\hat{u}_j^{\,p}(k+i-1\,|\,k) = \hat{u}_j^{\,p}(k+i-2\,|\,k) + \Delta\hat{u}_j^{\,p}(k+i-1\,|\,k)$$

Note that all the control inputs being optimized have a p superscript representing the current iteration. The other control inputs have a p-1 superscript and are the control input values for the other subsystems last iteration of the MPC problem. The optimal equations can then be rewritten

$$\frac{\partial V(k)}{\partial \hat{x}(k+i\,|\,k)} = C^T Q(i)\left(C\hat{x}(k+i\,|\,k)\right) + f(i) - Ip(i) + A^T p(i+1) + C_c{}^T \sigma(i) = 0 \qquad [77]$$

$$\frac{\partial V(k)}{\partial \Delta\hat{u}_j^{\,p}(k+i-1\,|\,k)} = R_j(i)\Delta\hat{u}_j^{\,p}(k+i-1/k) + I_j q_j(i) + E_j{}^T \pi_j(i) = 0 \qquad [78]$$

$$\frac{\partial V(k)}{\partial \hat{u}_j^{\,p}(k+i-1)} = B_j{}^T p(i) + I_j q_j(i+1) - I_j q_j(i) + F_j{}^T \lambda_j(i) = 0 \qquad [79]$$

$$A\hat{x}(k+i-1\,|\,k) + B_j \hat{u}_j^{\,p}(k+i-1\,|\,k) + \sum_{d=1,d\neq j}^{N_d} B_d \hat{u}_d(k+i-1\,|\,k) - \hat{x}(k+i\,|\,k) = 0 \qquad [80]$$

$$\hat{u}_j^{\,p}(k+i-2\,|\,k) + \Delta\hat{u}_j^{\,p}(k+i-1\,|\,k) - \hat{u}_j^{\,p}(k+i-1\,|\,k) = 0 \qquad [81]$$

$$E_j \Delta\hat{u}_j^p(k+i-1\,|\,k) - \Pi^{-1}{}_j(i)T_{1j}(i)\pi_j(i) - Y_{1j} = 0 \quad [82]$$

$$F_j \hat{u}_j^p(k+i-1\,|\,k) - \Lambda^{-1}{}_j(i)T_{2j}(i)\lambda_j(i) - Y_{2j} = 0 \quad [83]$$

$$C_c \hat{x}(k+i\,|\,k) + \Sigma^{-1}(i)T_3(i)\sigma(i) - Y_3 = 0 \quad [84]$$

Written in matrix form of $H_j\Theta_j = R_j$

$$\Theta_j = \begin{bmatrix} \vdots \\ \hat{u}_j^p(k+i-2\,|\,k) \\ \Delta\hat{u}_j^p(k+i-2\,|\,k) \\ \hat{x}(k+i-1\,|\,k) \\ p(i) \\ q_j(i) \\ \pi_j(i) \\ \lambda_j(i) \\ \sigma(i) \\ \hat{u}_j^p(k+i-1\,|\,k) \\ \Delta\hat{u}_j^p(k+i-1\,|\,k) \\ \hat{x}(k+i\,|\,k) \\ p(i+1) \\ q_j(i+1) \\ \vdots \end{bmatrix} \quad [85]$$

$$H_j = \begin{bmatrix} \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_j & E_j^T & 0 & 0 & 0 & R_j & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & B_j^T & -I_j & 0 & F_j^T & 0 & 0 & 0 & 0 & 0 & I_j & 0 \\ 0 & 0 & 0 & 0 & -I & 0 & 0 & 0 & C_C^T & 0 & 0 & C^T wQC & A^T & 0 & 0 \\ 0 & 0 & 0 & A & 0 & 0 & 0 & 0 & 0 & B_j & 0 & -I & 0 & 0 & 0 \\ 0 & I_j & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -I_j & I_j & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\Pi_j^{-1}T_{1j} & 0 & 0 & 0 & E_j & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Lambda_j^{-1}T_{2j} & 0 & F_j & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Sigma^{-1}T_3 & 0 & 0 & C_c & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots \end{bmatrix} \quad [86]$$

$$R_j = \begin{bmatrix} \vdots \\ 0 \\ 0 \\ -f(i) \\ -\sum_{\substack{d=1 \\ d \neq j}}^{N_d} B_d \hat{u}_d(k+i-1|k) \\ 0 \\ Y_{1j} \\ Y_{2j} \\ Y_3 \\ \vdots \end{bmatrix} \qquad [87]$$

Table 4 below summarizes the dimensions for each subsystem optimization problem for feasible cooperative MPC. Note that for a highly interactive system, all the subsystem states need to be included in the optimization to ensure a feasible answer. Depending on the number of control inputs, the computational burden for feasible cooperative MPC is larger than for multiplexed MPC. However when compared with centralized MPC, the computational burden is smaller. There is an iteration penalty for this algorithm that is negative in terms of computational burden, but it is necessary to achieve acceptable control performance.

Table 4: Feasible Cooperative MPC Dimension

Vector	Dimension
$\hat{x}(k+i-1\|k)$	N_x
$p(i)$	N_x
$q_j(i)$	$N_{u,j}$
$\pi_j(i)$	$N_{\Delta\hat{u}c}*N_{u,j}$
$\lambda_m(i)$	$N_{\hat{u}c}*N_{u,j}$
$\sigma(i)$	N_{zc}
$\hat{u}_j^p(k+i-1\|k)$	$N_{u,j}$
$\Delta\hat{u}_j^p(k+i-1\|k)$	$N_{u,j}$
Bandwidth Dimension	$(3 + N_{\Delta\hat{u}c} + N_{\hat{u}c})N_{u,j} + 2N_x + N_{zc}$

As the system becomes less interactive, feasible cooperative MPC provides the framework for significant reduction in computational burden. In addition to the control inputs, the state and output vectors can be broken up according the subsystems of the system.

$$\hat{x}(k+i-1\,|\,k)=\begin{bmatrix}\hat{x}_1(k+i-1\,|\,k)\\ \vdots \\ \hat{x}_{N_d}(k+i-1\,|\,k)\end{bmatrix} \qquad \text{for} \qquad i=1...H_p \qquad [88]$$

$$\hat{z}(k+i-1\,|\,k)=\begin{bmatrix}\hat{z}_1(k+i-1\,|\,k)\\ \vdots \\ \hat{z}_{N_d}(k+i-1\,|\,k)\end{bmatrix} \qquad \text{for} \qquad i=1...H_p \qquad [89]$$

The state dynamic equations then can be written in terms of the interactions between the subsystems.

$$\begin{aligned}\hat{x}_j(k+i\,|\,k)&=A_{jj}\hat{x}_j(k+i-1\,|\,k)+\sum_{d=1,d\neq j}^{N_d}A_{jd}\hat{x}_d(k+i-1\,|\,k)\\ &+B_{jj}\hat{u}_j(k+i-1\,|\,k)+\sum_{d=1,d\neq j}^{N_d}B_{jd}\hat{u}_d(k+i-1\,|\,k)\end{aligned} \qquad [90]$$

$$\hat{z}_j(k+i\,|\,k)=C_{jj}\hat{x}_j(k+i-1\,|\,k)+\sum_{d=1,d\neq j}^{N_d}C_{jd}\hat{x}_d(k+i-1\,|\,k) \qquad [91]$$

In these equations, the matrices relating control inputs, states, and system outputs are also broken down by subsystem. The off diagonal terms contain the interactions between the subsystems.

$$A=\begin{bmatrix}A_{11} & \cdots & A_{1N_d}\\ \vdots & \ddots & \vdots \\ A_{N_d1} & \cdots & A_{N_dN_d}\end{bmatrix} \qquad [92]$$

$$B=\begin{bmatrix}B_{11} & \cdots & B_{1N_d}\\ \vdots & \ddots & \vdots \\ B_{N_d1} & \cdots & B_{N_dN_d}\end{bmatrix} \qquad [93]$$

$$C = \begin{bmatrix} C_{11} & \cdots & C_{1N_d} \\ \vdots & \ddots & \vdots \\ C_{N_d 1} & \cdots & C_{N_d N_d} \end{bmatrix} \quad [94]$$

As the interactions between the subsystems become negligible, the off-diagonal terms in the A, B, and C matrices approach zero. These zeroes can be propagated throughout the MPC equations eliminating all the all other subsystem terms from the equations.

$$\frac{\partial V(k)}{\partial \hat{x}_j(k+i\,|\,k)} = C_{jj}{}^T Q_j(i)\left(C_{jj}\hat{x}_j(k+i\,|\,k)\right) + f_j(i) - I_j p_j(i) + A_{jj}{}^T p_j(i+1) + C_{cj}{}^T \sigma_j(i) = 0 \quad [95]$$

$$\frac{\partial V(k)}{\partial \Delta \hat{u}_j^p(k+i-1\,|\,k)} = R_j(i)\Delta \hat{u}_j^p(k+i-1/k) + I_j q_j(i) + E_j{}^T \pi_j(i) = 0 \quad [96]$$

$$\frac{\partial V(k)}{\partial \hat{u}_j^p(k+i-1)} = B_{jj}{}^T p_j(i) + I_j q_j(i+1) - I_j q_j(i) + F_j{}^T \lambda_j(i) = 0 \quad [97]$$

$$A_{jj}\hat{x}_j(k+i-1\,|\,k) + B_{jj}\hat{u}_j^p(k+i-1\,|\,k) - \hat{x}_j(k+i\,|\,k) = 0 \quad [98]$$

$$\hat{u}_j^p(k+i-2\,|\,k) + \Delta \hat{u}_j^p(k+i-1\,|\,k) - \hat{u}_j^p(k+i-1\,|\,k) = 0 \quad [99]$$

$$E_j \Delta \hat{u}_j^p(k+i-1\,|\,k) - \Pi^{-1}{}_j(i) T_{1j}(i)\pi_j(i) - Y_{1j} = 0 \quad [100]$$

$$F_j \hat{u}_j^p(k+i-1\,|\,k) - \Lambda^{-1}{}_j(i) T_{2j}(i)\lambda_j(i) - Y_{2j} = 0 \quad [101]$$

$$C_{cj}\hat{x}_j(k+i\,|\,k) + \Sigma_j{}^{-1}(i) T_{3j}(i)\sigma_j(i) - Y_{3j} = 0 \quad [102]$$

The matrix form of the non-interactive feasible cooperative MPC equations becomes the following.

$$\Theta_j = \begin{bmatrix} \vdots \\ \hat{u}_j(k+i-2 \mid k) \\ \Delta\hat{u}_j(k+i-2 \mid k) \\ \hat{x}_j(k+i-1 \mid k) \\ p_j(i) \\ q_j(i) \\ \pi_j(i) \\ \lambda_j(i) \\ \sigma_j(i) \\ \hat{u}_j(k+i-1 \mid k) \\ \Delta\hat{u}_j(k+i-1 \mid k) \\ \hat{x}_j(k+i \mid k) \\ p_j(i+1) \\ q_j(i+1) \\ \vdots \end{bmatrix} \qquad [103]$$

$$H_j = \begin{bmatrix} \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_j & E_j^T & 0 & 0 & 0 & R_j & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & B_{jj}^T & -I_j & 0 & F_j^T & 0 & 0 & 0 & 0 & 0 & I_j & 0 \\ 0 & 0 & 0 & 0 & -I_j & 0 & 0 & 0 & C_{Cj}^T & 0 & 0 & C_{jj}^T Q_j C_j & A_{jj}^T & 0 & 0 \\ 0 & 0 & 0 & A_{jj} & 0 & 0 & 0 & 0 & 0 & B_{jj} & 0 & -I_j & 0 & 0 & 0 \\ 0 & I_j & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -I_j & I_j & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\Pi_j^{-1} T_{1j} & 0 & 0 & 0 & E_j & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Lambda_j^{-1} T_{2j} & 0 & F_j & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Sigma_j^{-1} T_{3j} & 0 & 0 & C_{cj} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots \end{bmatrix} \qquad [104]$$

$$R_j = \begin{bmatrix} \vdots \\ 0 \\ 0 \\ -f(i) \\ 0 \\ 0 \\ Y_{1j} \\ Y_{2j} \\ Y_{3j} \\ \vdots \end{bmatrix} \qquad [105]$$

Table 5 summarizes the dimensions for feasible cooperative control without interactions. It can be seen that the dimension on the terms associated with the state vector and the system level constraints are reduced to just the terms associated with the appropriate subsystem, thus reducing the overall computational burden. One other item of significance must be noted. Since there are no interactions, there are no iterations required for feasible cooperative multiplexed control to find the optimal converged solution. Therefore as the numbers of interactions are reduced, the computational burden of the MPC algorithm decreases also. It must be noted that as the interactions decrease, the computational burden of centralized MPC also becomes smaller. This implies that, there potentially could be a relationship between the "diagonality" of the A, B, and C matrices and the number of iterations required for an optimal solution of the MPC problem.

Table 5: Feasible Cooperative Dimension with no Interactions

Vector	Dimension
$\hat{x}(k+i-1\mid k)$	$N_{x,j}$
$p(i)$	$N_{x,j}$
$q_j(i)$	$N_{u,j}$
$\pi_j(i)$	$N_{\Delta\hat{u}c}*N_{u,j}$
$\lambda_m(i)$	$N_{\hat{u}c}*N_{u,j}$
$\sigma(i)$	$N_{zc,j}$
$\hat{u}_j^p(k+i-1\mid k)$	$N_{u,j}$
$\Delta\hat{u}_j^p(k+i-1\mid k)$	$N_{u,j}$
Bandwidth Dimension	$(3 + N_{\Delta\hat{u}c}+ N_{\hat{u}c})N_{u,j} + 2N_{x,j} + N_{zc,j}$

5.4 **Feasible Cooperative Multiplexed MPC**

Both multiplexed MPC and feasible cooperative control can be integrated together into a single algorithm that can combine the computational burden reduction of each method while alleviating some of the weaknesses of each.

$$\frac{\partial V(k)}{\partial \hat{x}(k+i\mid k)} = C^T Q(i)\left(C\hat{x}(k+i\mid k)\right) + f(i) - Ip(i) + A^T p(i+1) + C_c{}^T \sigma(i) = 0 \qquad [106]$$

$$\frac{\partial V(k)}{\partial \Delta \hat{u}^{p}_{j,m}(k+i-1\mid k)} = R_{j,m}(i)\Delta \hat{u}^{p}_{j,m}(k+i-1/k) + I_{j,m}q_{j,m}(i) + E_{j,m}{}^{T}\pi_{j,m}(i) = 0 \quad [107]$$

$$\frac{\partial V(k)}{\partial \hat{u}^{p}_{j,m}(k+i-1)} = B_{j,m}{}^{T}p(i) + I_{j,m}q_{j,m}(i+1) - I_{j,m}q_{j,m}(i) + F_{j,m}{}^{T}\lambda_{j,m}(i) = 0 \quad [108]$$

$$A\hat{x}(k+i-1\mid k) + B_{j,m}\hat{u}^{p}_{j,m}(k+i-1\mid k) + \sum_{d=1,d\neq j}^{N_d} B_{d,m}\hat{u}^{p-1}{}_{d,m}(k+i-1\mid k) - \hat{x}(k+i\mid k) = 0 \quad [109]$$

$$\hat{u}^{p}_{j,m}(k+i-2\mid k) + \Delta \hat{u}^{p}_{j,m}(k+i-1\mid k) - \hat{u}^{p}_{j,m}(k+i-1\mid k) = 0 \quad [110]$$

$$E_{j,m}\Delta \hat{u}^{p}_{j,m}(k+i-1\mid k) - \Pi^{-1}{}_{j,m}(i)T_{1\,j,m}(i)\pi_{j,m}(i) - Y_{1\,j,m} = 0 \quad [111]$$

$$F_{j,m}\hat{u}^{p}_{j,m}(k+i-1\mid k) - \Lambda^{-1}{}_{j,m}(i)T_{2\,j,m}(i)\lambda_{j,m}(i) - Y_{2\,j,m} = 0 \quad [112]$$

$$C_c\hat{x}(k+i\mid k) + \Sigma^{-1}(i)T_3(i)\sigma(i) - Y_3 = 0 \quad [113]$$

Where $\eta(k) = k \bmod(N_{u,j}) + 1$

$\Delta\hat{u}_{j,m}(k+i-1|k)$ only changes the $\eta(k)^{th}$ value of the $\Delta\hat{u}_j(k+i-1|k)$ vector
$\hat{u}_{j,m}(k+i-1|k)$ only changes the $\eta(k)^{th}$ value of the $\hat{u}_j(k+i-1|k)$ vector
$B_{j,m}$ is the $\eta(k)^{th}$ column vector of B_j

The matrix form of the feasible cooperative multiplexed MPC (FCMMPC) equations becomes the following. Written in the matrix form $H_{j,m}\Theta_{j,m} = R_{j,m}$

$$\Theta_{j,m} = \begin{bmatrix} \vdots \\ \hat{u}^{p}_{j,m}(k+i-2 \mid k) \\ \Delta\hat{u}^{p}_{j,m}(k+i-2 \mid k) \\ \hat{x}(k+i-1 \mid k) \\ p(i) \\ q_{j,m}(i) \\ \pi_{j,m}(i) \\ \lambda_{j,m}(i) \\ \sigma(i) \\ \hat{u}^{p}_{j,m}(k+i-1 \mid k) \\ \Delta\hat{u}^{p}_{j,m}(k+i-1 \mid k) \\ \hat{x}(k+i \mid k) \\ p(i+1) \\ q_{j,m}(i+1) \\ \vdots \end{bmatrix} \qquad [114]$$

$$H_{j,m} = \begin{bmatrix} \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{j,m} & E_{j,m}^{T} & 0 & 0 & 0 & R_j & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & B_{j,m}^{T} & -I_{j,m} & 0 & F_{j,m}^{T} & 0 & 0 & 0 & 0 & 0 & I_{j,m} & 0 \\ 0 & 0 & 0 & 0 & -I & 0 & 0 & 0 & C_C^{T} & 0 & 0 & C^{T}wQCA^{T} & 0 & 0 & \\ 0 & 0 & 0 & A & 0 & 0 & 0 & 0 & 0 & B_{j,m} & 0 & -I & 0 & 0 & 0 \\ 0 & I_{j,m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -I_{j,m} & I_{j,m} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\Pi^{-1}_{j,m}T_{1j,m} & 0 & 0 & 0 & E_{j,m} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Lambda^{-1}_{j,m}T_{2j,m} & 0 & F_j & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Sigma^{-1}T_3 & 0 & 0 & C_c & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots \end{bmatrix} \qquad [115]$$

$$R_{j,m} = \begin{bmatrix} \vdots \\ 0 \\ 0 \\ -f(i) \\ -\sum_{\substack{d=1 \\ d \neq j}}^{N_d} B_{d,m}\hat{u}_{d,m}(k+i-1 \mid k) \\ 0 \\ Y_{1\,j,m} \\ Y_{2\,j,m} \\ Y_3 \\ \vdots \end{bmatrix} \qquad [116]$$

Table 6 below summarizes the dimensions for each subsystem optimization problem for feasible cooperative multiplexed MPC. Note that the dimension for this algorithm is almost identical to multiplexed MPC with the added penalty of the iterations. However each subsystems will be multiplexing $N_{u,j}$ inputs as opposed to N_u inputs for a centralized MPC algorithm. Thus given the same control sampling rate, the feasible cooperative multiplexed MPC algorithm will most likely result in a better control performance than multiplexed MPC and more comparable to centralized MPC. Additionally due to the distributed framework of FCMMPC, as the interactions of the system become smaller, the dimension of the problem potentially becomes smaller.

Table 6: Feasible Cooperative Multiplexed MPC Dimensions

Vector	Dimension
$\hat{x}(k+i-1\mid k)$	N_x
$p(i)$	N_x
$q_{j,m}(i)$	1
$\pi_{j,m}(i)$	$N_{\Delta\hat{u}c}$
$\lambda_{j,m}(i)$	$N_{\hat{u}c}$
$\sigma(i)$	N_{zc}
$\hat{u}_{j,m}(k+i-1\mid k)$	1
$\Delta\hat{u}_{j,m}(k+i-1\mid k)$	1
Bandwidth Dimension	$3 + N_{\Delta\hat{u}c} + N_{\hat{u}c} + 2N_x + N_{zc}$

5.5 Tip-Jet Reaction Drive MPC Computational Burden Reduction

For the tip-jet reaction drive system there at a minimum of six control inputs. As will be seen in the following chapters, there is a minimum of eight states in system and the prediction horizon of 100 steps. The computational burden reduction can be analyzed using the bandwidth dimensions of the centralized MPC and feasible cooperative multiplexed MPC defined in Table 2 and Table 6, respectively, Assuming there are no constraints and given the defined dimensions, the computational burden is reduced by a factor of six by using the feasible cooperative multiplexed MPC. As the number of

constraints on the output parameters is increased, the benefits seen by incorporating the feasible cooperative multiplexed are reduced to a factor closer to three if there are 20 constraints present. However, since the feasible cooperative multiplexed MPC only optimizes a single control input at a time, the number of control input position and rate constraints present on each MPC are reduced by a factor of six. This may result in almost an order of magnitude reduction in the computational burden by using the proposed approach.

Given this understanding of the computational burden benefit by using the proposed MPC on the tip-jet reaction drive system, the performance of the proposed controller needs to be understood. To provide a basis of comparison, the performance of the proposed controller will be compared with both a PI controller and a centralized MPC. In the next three chapters, a PI controller, a centralized MPC, and a feasible cooperative multiplexed MPC for the tip-jet reaction drive system will be defined and sized. Once sized, the performance of each controller can be captured and compared. If the performance of the feasible cooperative multiplexed MPC is similar to that of the centralized MPC, the hypothesis of the disseration will have been confirmed.

CHAPTER 6

BASELINE PI CONTROLLER

Before the proposed MPC architecture performance is analyzed, a baseline controller needs to be defined, sized, and analyzed. A PI controller was chosen as the baseline because of its simplicity and wide application on a number of aerospace systems with similar components to a tip-jet reaction drive system. This chapter is broken down into 5 sections. The first section will provide a general overview of the PI controller design process. The second section will explore and define different components of the baseline PI control architecture. In the third section the PI controller will be sized and its performance and stability properties will be analyzed. The fourth section will simulate different engine throttle and rotor load demand changes to capture the transient performance of the control. And the last section will contain a brief summary and draw conclusions from the previous sections.

6.1 **Baseline PI Controller Design Process**

Figure 14 below shows a general step-by-step process that was followed to create the baseline PI controller. In general, the PI controller design process can be broken up into three general categories: controller architecture components definition and modeling, controller design and sizing, and transient simulations. Within each of these general categories are multiple tasks.

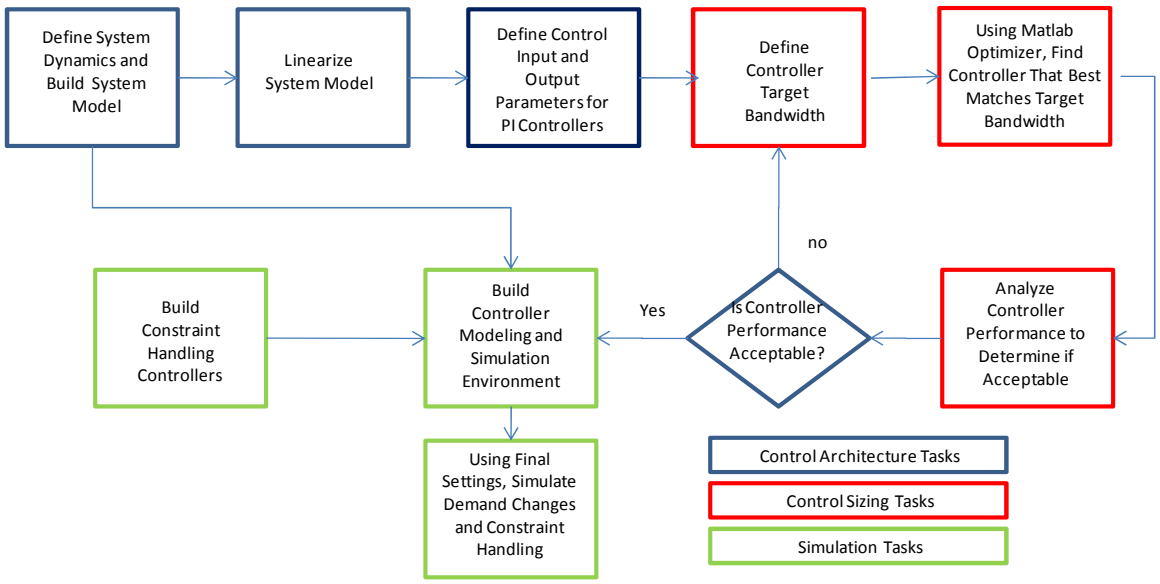

Figure 14: PI Controller Design Process

The first controller design category has three tasks. The first task is to define and quantify the dynamics of the system and build a non-linear system model of the tip-jet reaction drive system. This task is described in detail in section 6.2.1. The second task is to linearize the tip-jet reaction drive system model. Although not covered in detail in this dissertation, this allows for the use of many different frequency based design techniques. The third task is to define the appropriate control input and output parameter. This task uses both the non-linear and linear models. The details of this task are discussed in detail in section 6.2.2. Once these three tasks are done the controller can be sized.

Using the linear model developed in the previous category, the PI controller can be sized and its performance can be analyzed. The first task in the controller sizing category is to define the target bandwidth of the controller. For this dissertation, the target bandwidth was defined using the system dynamics outlined in the previous category. The second task, which is described in 6.3.1, is to use the Edmund's algorithm to size the controller. Once the controller is sized, the last task is to analyze the performance of the controller. The details of this task are discussed in section 6.3.2. If

the controller does not have acceptable performance properties, the target bandwidth needs to be redefined.

Once a controller with acceptable performance qualities has been sized, various transient simulations can be analyzed. However, before any transient simulations can be run, the modeling and simulation (M&S) environment of the controller has to be created. This task is discussed in section 6.4.1. Using the M&S environment, simulations varying both throttle demand and rotor load changes can be run. The results of the simulations and discussion of the results is contained in section 6.4.2. The last simulation task to be performed is demonstration of PI controller constraint handling. Before this simulation is run, a controller specific to that constraint has to be designed. The details of this task are covered in section 6.4.3. Once all these tasks are complete, the performance capabilities of a PI controller are understood and comparisons with other controllers can be made.

6.2 **Baseline PI Control Architecture**

Figure 15 below shows the baseline PI control architecture. The two main components of the architecture are the physical system and the PI controller which are defined by G(s) and K(s) respectively. The tip-jet reaction drive system, G(s), is composed of coupled nonlinear rotor, heat soak, and gas path dynamics, which occur at multiple frequency scales. The controller translates the comparison of a reference, r(s), signal with measured system output, y(s), into inputs to the physical system, u(s). The PI controller, K(s) contains a six by six matrix of proportional and integral gains.

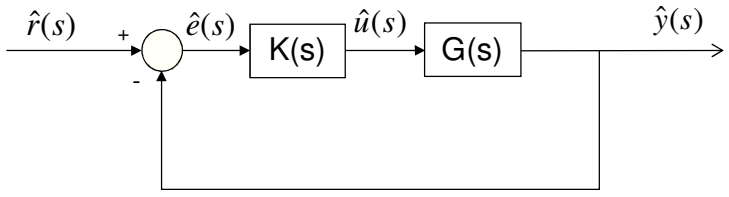

Figure 15: Baseline PI Controller Architecture

A couple of areas need to be explored and understood before the PI controller gains are sized. First, the dynamics of the system need to be quantified. Not only will this help size the gains, it will also define the sampling interval required for the control and provide a solid understanding of the response of the system. After quantifying the dynamics of the tip-jet reaction drive system, it is necessary to discuss the selection of the inputs and outputs used in the control. Relative gain array and probabilistic analysis of the system performance over the useful life will be used to determine the appropriate input/output combination.

6.2.1 System Dynamics Definition

One of the first tasks will be to develop a model of the tip-jet reaction drive system that captures the dynamics of the system. Tai explored the steady state design and sizing of different reaction drive systems (TAI 1998). Kong, Park, and Kang studied the transient performance of the canard rotor wing (CRW) system modeling both the rotor and mass conservation dynamics (KONG, PARK and KANG 2006). In an engine system transient, there are three general types of dynamics: rotor, heat soak, and gas path (KOPASAKIS, et al. 2008). The rotor dynamics, equation 117, are often the slowest and most dominant of the dynamics and must be included in any system transient simulation. The gas path dynamics (often called volume dynamics), which are quantified by the unsteady mass, momentum, and energy conservation equations 119-121, are generally the fastest.

Rotor Dynamics:

$$\frac{dN_i}{dt} = \frac{\sum T}{J_i} \qquad [117]$$

Heat Soak Dynamics

$$\frac{dT_{metal_i}}{dt} = \frac{hA_{hx}(T_{gas} - T_{metal_i})}{m_i c_{p_i}} \qquad [118]$$

Gas Path (Volume) Dynamics

Mass Conservation

$$\frac{d\rho_i}{dt} = \frac{\sum W}{V_i} \qquad [119]$$

Momentum Conservation

$$\frac{dW_i}{dt} = \frac{force + \sum PA}{L_i} \qquad [120]$$

Energy Conservation

$$\frac{dP_i}{dt} = \frac{power + \sum Wh}{V_i} \qquad [121]$$

Lower frequency models (~50Hz), such as NPSS, only include rotor and heat soak dynamics (LYTLE 2000). Higher frequency (dynamic) models (~2500 Hz) require the incorporation of all the dynamics. This is especially necessary during the simulation of high frequency events such as an engine stall, inlet temperature spikes, afterburner light off, or component failures (KHALID and HEARNE 1980). Even if the aforementioned events were not present, the presence of relatively large ducts alone may cause the gas path dynamics to be non-negligible. Thus, it is imperative to include these equations in the system model. The dynamic model can then be used to define the minimum realization linear matrices for the tip-jet reaction drive system dynamics used in both subsequent MPC algorithms.

6.2.1.1 Volume Dynamics Modeling

A feasible way to incorporate the high frequency gas path dynamics is by using the one dimensional lumped control volume approach similar to methods developed in the mid 1980's (CHUNG, LEAMY and COLLINS 1985)(HOSNY and BITTER 1985). In this approach, each component (such as the fan, compressor, burner, or duct) may have a control volume associated with it as shown in Figure 16. The calculated terms (such as

pressure drop/gain, heat transfer, or energy loss/gain) in each component are input as force or power terms in each control volume. Referencing the tip-jet reaction drive system schematic in Figure 4, each element shown would have an associated control volume.

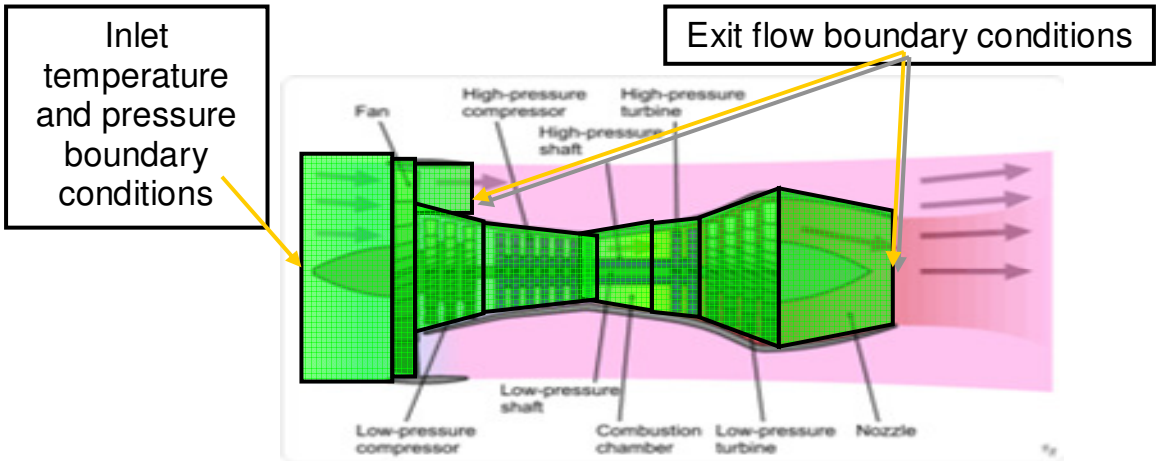

Figure 16: An Engine with Volume Dynamics

6.2.1.2 Tip-Jet Reaction Drive Dynamics

Given a typical control sampling rate of 20ms, the frequency range of interest is less than 50 rad/s. To accurately model the effect of the dynamics, a factor of approximately five is applied to the frequency rate of interest to determine which dynamics need to be analyzed. For a sampling rate of 20 ms, dynamics greater than approximately 10 to 20 rad/s can be neglected. Using moments of inertia and thermal masses of a typical engine (SKIRA and DEHOFF 1978) or rotorcraft (BALLIN 1988) and simple first order sensor and actuator models (FREDERICK, GARG and COLLINS 2000), the eigenvalues associated with the rotor, heat soak, sensor, and actuator dynamics are shown in Table 7. The rotor and heat soak dynamics are most closely associated with the smallest eigenvalues and in turn are the most dominant dynamic.

Table 7: Tip-Jet Eigenvalues and Associated Dynamics

Eigenvalues, rad/s	Associated State
-50	RPM sensor
-33.3333	Pressure sensor
-26.9363	Tip Burner
-26.001	Fuel Actuator
-18.0018	Exhaust Area Actuator
-6.7835	Engine Burner
-6.5698	Engine Burner
-3.0649+0.4018i	LP and HP Rotor Shaft
-3.0649-0.4018i	LP and HP Rotor Shaft
-3.0767	LP and HP Rotor Shaft
-2.4253	LP and HP Rotor Shaft
-0.4773	Tip Rotor Shaft

The control volume dimensions of the engines in the tip-jet reaction drive system are defined by using order of magnitude estimates based on the dimensions of a 3,000 lbs thrust class turbofan engine. Since the engines have a relatively high bypass ratio, the bypass ducts will have similar cross-sectional areas to that of the fan inlet. In order to fully understand the effect of the duct volumes on the response of the system different, duct volume dimensions estimates were studied. The different columns of Table 8 show how the top sixteen dominant eigenvalues change as the duct dimension is changed. The duct dimension in the first column is 10 times the size of a volume for the engine. The next two columns represent increases in duct dimension of 20 to 100 times the size of volume for an engine component. When the duct dimension is only 20 times the size of an engine component volume, only a couple of additional eigenvalues less than 50 rad/s appear. However these are still greater than the critical modeling frequency of 10-20 rad/s and therefore can be neglected. As the dimension increases, the number of eigenvalues less than 50 rad/s significantly increases, such that when the dimension is 100 times larger the additional eigenvalues approach the same order as the shaft and heat soak dynamics and need to be included in the model.

Table 8: Shaft, Heat Soak, Gas Path Eigenvalues with Duct Dimension Relative to Engine Dimension

10x, rad/s	20x, rad/s	100x, rad/s
-119.00	-56.50	-18.00
-119.00	-56.50	-15.20
-101.00	-50.00	-15.20
-101.00	-46.20	-12.40
-50.00	-46.20	-12.40
-33.30	-33.30	-10.70
-26.90	-26.90	-10.70
-26.00	-26.00	-8.73
-18.00	-18.00	-8.73
-7.38	-7.38	-7.41
-7.35	-7.36	-7.41
-3.18	-3.18	-3.19
-3.08	-3.08	-3.09
-1.52	-1.52	-1.52
-1.50	-1.50	-1.50
-0.49	-0.49	-0.49

Low frequency models that neglect gas path dynamics, such as NPSS, would be suitable for simulating a tip-jet reaction drive system with duct dimensions similar to the 10x column. However, as the dimensions approach the 100x column, the gas path dynamics would need to be integrated into the system model. For the purposes of this study, the dimensions defined in the 10x column will be used for the subsequent runs. But all the analysis below would be the same no matter which dimension was chosen since the system model has the gas path analysis integrated.

6.2.1.3 Tip-Jet Reaction Drive Modeling and Simulation Environment

Since the volume dynamics for the tip-jet reaction drive system can be neglected, the system was modeled using NPSS. NPSS is an object orientated code, with generic elements for different components such as compressors, burners, turbines, and shafts. The components comprising the tip-jet reaction drive NPSS model are very similar to those shown in Figure 4 although the actual model has both a low pressure (LP) and high

pressure (HP) shaft. The LP shaft connects the fan and LP turbine while the HP shaft connects the compressor and HP turbine.

The components of the tip-jet reaction drive can be grouped into three general subsystems: two engines and one tip-jet reaction drive subsystem. Although physically two separate subsystems, the components used to model each engine subsystem are identical. Figure 17 below shows the functional component layout used to model each engine subsystem. The engine subsystem is a mixed flow turbofan where fan air is split into core airflow which combusted in the burner and bypass air which bypasses the burner. When the tip-jet reaction drive is operating in hover mode, virtually all of the bypass air is ducted to the tip-jet reaction drive system. The core airflow is ducted through an exhaust nozzle to generate a thrust which can be used to help control the vehicle using the tip-jet reaction drive system.

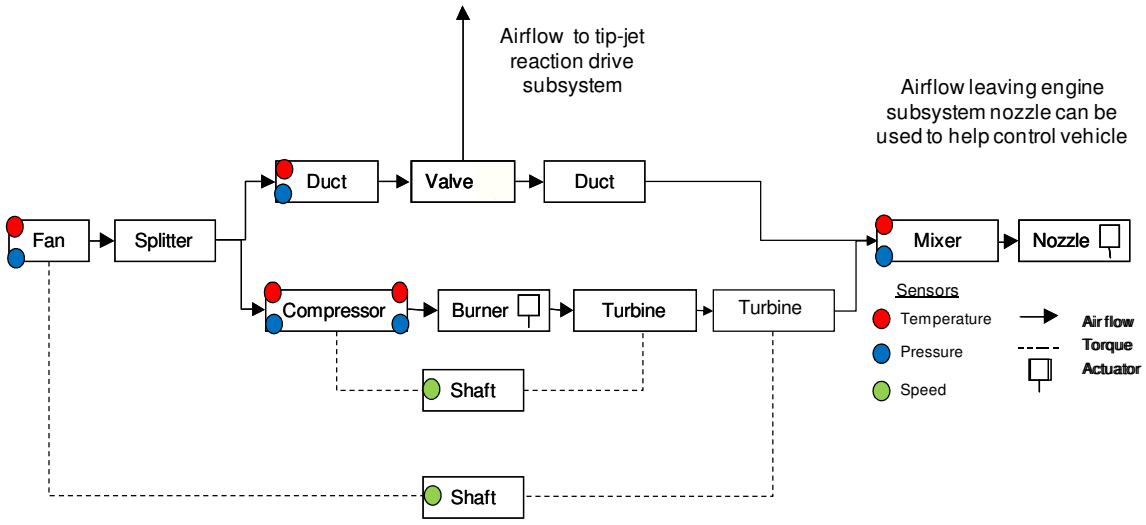

Figure 17: Tip-Jet Reaction Drive Engine Subsystem Component Model

In hover mode, the cold bypass air of the two engines enters into rotor hub at plenum component in Figure 18. It is then routed out of each of the rotor blades to the tip-jet. Duct components are included to provide sources of pressure loss through the reaction drive subsystem. At the tip of each blade, fuel is mixed with the air and combusted. The hot combustion air is then accelerated through exhaust nozzles at the tip

of each blade thus providing the reaction torque for the rotor, and in turn the necessary lift and thrust required for the rotor system.

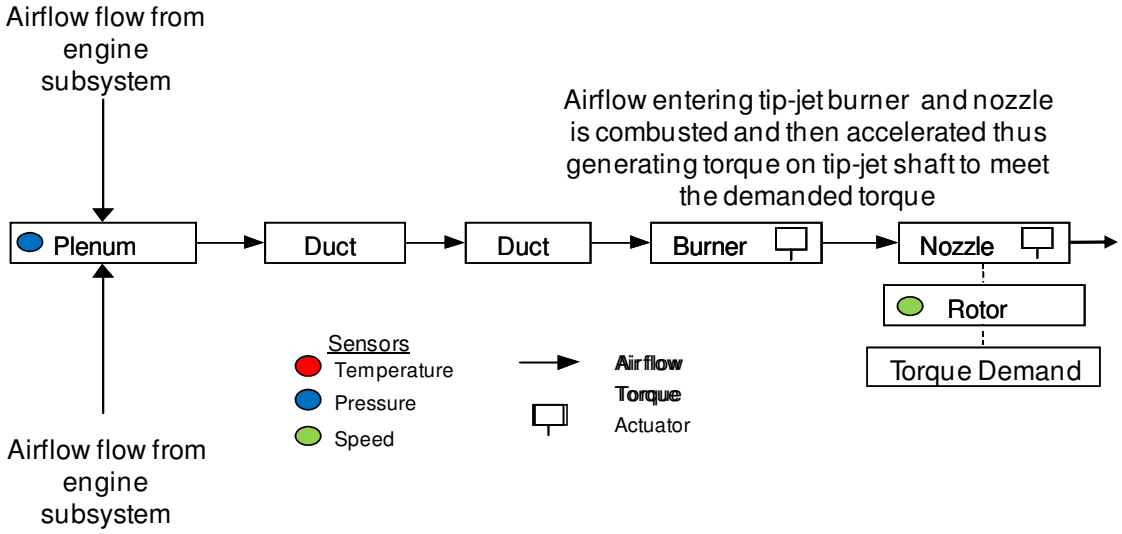

Figure 18: Tip-Jet Reaction Drive Subsystem Component Model

The torque required for the tip-jet reaction drive subsystem is modeled as a time varying "black box" variable thus requiring no knowledge of blade aero. Flow to each tip-jet rotor blade assumed to be split equally and therefore modeled as single "duct", "burner", and "nozzle" components. In addition, since the tip-jets are located on a rotating surface, the amount of torque generated by the tip-jets is proportional to the difference between the air velocity exiting the tip-jets and the actual blade velocity. This relationship is described below where T, w, and v represent torque, mass flow, and velocity, respectively.

$$T_{tip-jet} = w_{tip-jet}(v_{tip-jet} - v_{blade\_tip}) \qquad [122]$$

Both velocities in the above equation are relative to the rotor blade hub. The shaft dynamics equation for the tip-jet reaction drive shaft can then be written as follows.

$$\frac{dN_{tip-jet\_shaft}}{dt} = \frac{T_{tip-jet} - T_{required}}{J_{tip-jet}} \qquad [123]$$

The two engine subsystem models and tip-jet reaction drive subsystem model are combined together into a single integrated NPSS model as detailed in Appendix A. This model will represent the tip-jet reaction drive system in the control design work in all the subsequent chapters. Once this model of the tip-jet reaction drive system is built, the control design and sizing process for the PI controller can begin.

6.2.2 Control Input and Output Selection and Steady State Performance

The next step in developing control laws is to define which target variables are to be controlled, i.e., the purpose of the controller. In general, the fuel metering valves are used to control thrust or lift, while the exhaust valves are used to ensure that operability or performance limits are not exceeded. However since thrust or stall margin cannot be measured, certain measurements that correlate well with them are substituted. For thrust, typically the LP or HP shaft speed correlate well. To control the stall margin or the operating line of the system, either engine pressure ratio (EPR) or core engine pressure ratio (CEPR) is used. For the tip-jet subsystem tip-jet shaft speed is a natural choice for a control parameter. Tip-jet nozzle EPR or bypass duct pressure both present viable options for a pressure ratio control. Additionally, there are a couple more general metrics used to select which variables are to be controlled: 1.) minimal interactions between inputs and outputs and 2.) minimal variance of unmeasured parameters of interest over the functional life of the system.

6.2.2.1 Relative Gain Array

The relative gain array (RGA) is a very simple and useful measure to capture the sensitivities and interactions between different potential input/output combinations (BRISTOL 1966). Table 9 below shows a relative gain array for the tip-jet all the potential input/output combinations for the tip-jet reaction drive system at a frequency of

0 rad/s (i.e., steady-state). Other RGAs generated at different frequencies are located in Appendix B. In general, for frequencies of interest, the overall trends of different RGA tables are fairly similar. A value of 1 in the table represents a significant relationship between input and output and the converse holds true for a value of zero. To minimize interactions, the choice of the final input/output combination should be as diagonally dominant as possible (i.e., the input/output value should be as close to 1 as possible).

Table 9: Relative Gain Array of Tip-Jet Reaction Drive System at 0 rad/s

Relative Gain Array	Right Fuel Flow	Right Exhaust Area	Left Fuel Flow	Left Exhaust Area	Tip-Jet Fuel Flow	Tip-Jet Exhaust Area
LP Shaft Right	**0.356**	**0.204**	0.000	0.000	0.011	0.070
HP Shaft Right	**0.292**	0.027	0.000	0.000	0.006	0.019
EPR Right	**0.250**	**0.260**	0.000	0.000	-0.017	0.062
CEPR Right	0.040	**0.480**	0.000	0.000	-0.034	0.081
LP Shaft Left	0.000	0.000	**0.355**	**0.205**	0.011	0.070
HP Shaft Left	0.000	0.000	**0.292**	0.027	0.006	0.019
EPR Left	0.000	0.000	**0.251**	**0.259**	-0.017	0.061
CEPR Left	0.001	0.000	0.040	**0.480**	-0.034	0.081
Tip Jet Shaft	0.001	-0.001	0.001	-0.002	**1.373**	**-0.372**
EPR Tip	0.016	0.007	0.015	0.007	**-0.106**	**0.321**
Duct Pressure Right	0.044	0.025	0.002	-0.001	**-0.100**	**0.293**
Duct Pressure Left	0.002	-0.001	0.043	0.025	**-0.100**	**0.293**

From Table 9, the PI controller naturally breaks into 3 distinct 2x2 input/output pairings. The tip-jet fuel flow and exhaust area control inputs have large interactions with tip-jet shaft speed, tip-jet nozzle EPR, and bypass duct pressures. For reasons of symmetry, tip-jet EPR was chosen as the pressure ratio to be controlled. The fuel flow and exhaust area on each engine has the most significant interactions with the LP shaft speed, HP shaft speed, EPR, and CEPR of that engine. However, from the RGA tables there are no obvious reasons to down select the two most appropriate input/output control pairings for each engine. The steady state performance variation over the life of the system of potential engine control input/output pairings will further aid in selecting the most appropriate input/output pairing for the engines.

6.2.2.2 Steady State Performance Variation

Before any final choices on control input/output combinations are made, the variance of unmeasured performance metrics over the life of the system need to be analyzed. Unmeasured parameters like thrust and stall margin are indirectly controlled via a correlation with a measured parameter. This correlation may change over the useable life of the system which may result in margining of both steady-state and transient performance. The input/output control pair should be chosen such that variance of unmeasured parameters is minimal over the useable life of the system (BROWN and ELGIN 1985). Table 10 summarizes some of the variance of performance metrics such as engine thrust, turbine inlet temperature, and compressor stall margin over the life of the engine with different engine control input/output pairings. Note that the HP shaft speed and EPR pair is not included because of significant convergence errors in the simulation.

Table 10: Steady State Performance Variation for Different Input/Output Pairings

		LP and CEPR		HP and CEPR		LP and EPR	
Parameter	Age	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
Engine Thrust	New	0.999	0.008	1.001	0.021	0.999	0.011
TSFC	New	1.004	0.014	1.005	0.032	1.004	0.014
LP Shaft Speed	New	1.000	0.000	1.007	0.025	1.000	0.000
HP Shaft Speed	New	1.000	0.013	1.000	0.000	1.000	0.013
Turbine Inlet Temperature	New	1.004	0.012	1.005	0.030	1.004	0.012
LP Stall Margin	New	1.000	0.008	0.097	0.145	1.000	0.008
HP Stall Margin	New	0.983	0.076	0.997	0.086	0.984	0.076
Engine Thrust	Degraded	1.008	0.011	1.006	0.021	1.008	0.011
TSFC	Degraded	1.040	0.014	1.037	0.031	1.040	0.014
LP Shaft Speed	Degraded	1.000	0.000	1.003	0.024	1.000	0.000
HP Shaft Speed	Degraded	1.002	0.013	1.000	0.000	1.002	0.013
Turbine Inlet Temperature	Degraded	1.041	0.011	1.038	0.031	1.041	0.012
LP Stall Margin	Degraded	1.002	0.007	0.948	0.152	1.002	0.007
HP Stall Margin	Degraded	1.049	0.075	1.065	0.086	1.049	0.075

In this analysis it is clear that LP stall margin was significantly reduced over the life of the system when HP shaft speed was used in the control as opposed to LP shaft speed. This was the overriding factor in choosing LP shaft speed over HP shaft speed.

There is minimal difference between using EPR or CEPR control. CEPR was chosen as the pressure control mainly because of its single large RGA value; however, EPR appears to be a perfectly feasible choice. Now that the control input/output parameters are defined and the M&S environment has been developed, the PI controller can be sized.

6.3 **Control Sizing and Analysis**

Since the tip-jet reaction drive system has three fairly distinct subsystems (2 engines and 1 reaction drive rotor), the PI controller for the system can be broken down into three smaller distributed subsystem controllers. The merits of using either a centralized or distributed controller need to be discussed before sizing the PI controller gains. A centralized controller will account for all the interactions between inputs and outputs. In terms of performance, this would be the most optimal setup. However, additional costs may be incurred when accounting for communication needs over the large distance between control inputs and outputs; this makes a centralized controller a less attractive option. If the interactions between the subsystems are small enough, a distributed controller may provide a feasible option. Besides the cost argument, an additional argument for a distributed controller is based on the RGA analysis from the previous section which naturally broke the control input/output relationships into three distinct subsystems. Based on these two arguments, the distributed PI controller for each subsystem will be studied.

Given the dynamics of the system defined in the previous section, the target open loop bandwidth for both the LP shaft speed and CEPR response of the engine subsystem controls were chosen to be 3 rad/s. Similarly, the tip-jet rotor shaft speed target bandwidth was chosen to be 0.6 rad/s. Since the hub pressure is more of a safety metric than a performance metric and it correlates well with engine stall margin, the target bandwidth for hub pressure was chosen to be the same as that of the engine components.

6.3.1 Control Sizing Methodology

Using the Edmunds algorithm, a set of PI gains will be defined for each of the distributed subsystem controller. The control will then be sized to match a target bandwidth performance. Bandwidth is a frequency based metric which defines the maximum frequency in which the output of the system is affected by inputs to the system. For a given sinusoidal input signal, the magnitude (or gain) of the response is defined as the ratio of the magnitude of the output signal to that of the input signal. An output signal with a magnitude of less than 70% the input signal is a generally accepted threshold for closed loop bandwidth (SKOGESTAD and POSTLETHWAITE 2005). For use in Bode analysis, this ratio is converted to decibels (dB) by taking the log of the ratio and multiplying by 20, resulting is a closed loop bandwidth magnitude of -3 dB.

6.3.2 Control Response Analysis

Bode plots provide an excellent framework for analyzing controller performance by capturing both the magnitude of the response of the system outputs relative to the magnitude of the inputs and the change in phase angle of the outputs relative to the inputs. These plots will be used throughout the next few chapters to provide a quantitative comparison between the different control architectures. The specific performance metrics of interest are bandwidth, interactions, gain margin, and phase margin. On each magnitude and phase plot there are 36 subplots representing the magnitude or phase of a response to a given demand change. Each column in the Bode plot represents the change in demand, whereas each row represents the response of the parameter to a change in demand. To keep the description of the plots simple, shorthand was used for the subplot descriptions. Table 11 below summarizes the shorthand descriptions used for the PI controller Bode plot. To demonstrate the shorthand notation, the upper left subplot in magnitude Bode plot in Figure 19 can be used. In this subplot, the column description is “N2 Right” and the row description is “N2 Right”. Thus the

response in the figure represents the actual magnitude of the LP shaft speed response to changes to a LP shaft speed demand. When describing the plots in the following chapters, this shorthand notation will be used

Table 11: PI Controller Bode Plot Descriptions

Bode Plot Column and Row Description	Column Description	Row Description
N2 Right	Right engine LP shaft speed demand change	Right engine LP shaft speed response to a demand change
CEPR Right	Right engine CEPR demand change	Right engine CEPR response to a demand change
N2 Left	Left engine LP shaft speed demand change	Left engine LP shaft speed response to a demand change
CEPR Left	Left engine CEPR demand change	Left engine CEPR response to a demand change
Tip-jet Shaft	Tip-jet shaft speed demand change	Tip-jet shaft speed response to a demand change
Hub Pressure	Hub pressure demand change	Hub pressure response to a demand change

Figure 19 and Figure 20 below show Bode plots of the closed loop magnitude and phase, respectively, of the PI controller. The diagonal terms in Figure 19 can be used to capture the bandwidth of the controller. The bandwidth is described as the frequency at which the magnitude is -3 dB. For the remaining Bode plots included in this dissertation, the green circle highlights delineate the subplots that represent the bandwidth. Another metric for good control performance is minimal interactions between the off-diagonal terms. In Figure 19 the largest magnitude for any off-diagonal term is approximately -10 dB, which demonstrates that there are minimal interactions for closed loop operation. For the remainder of this dissertation the largest interactions will be highlighted using red squares.

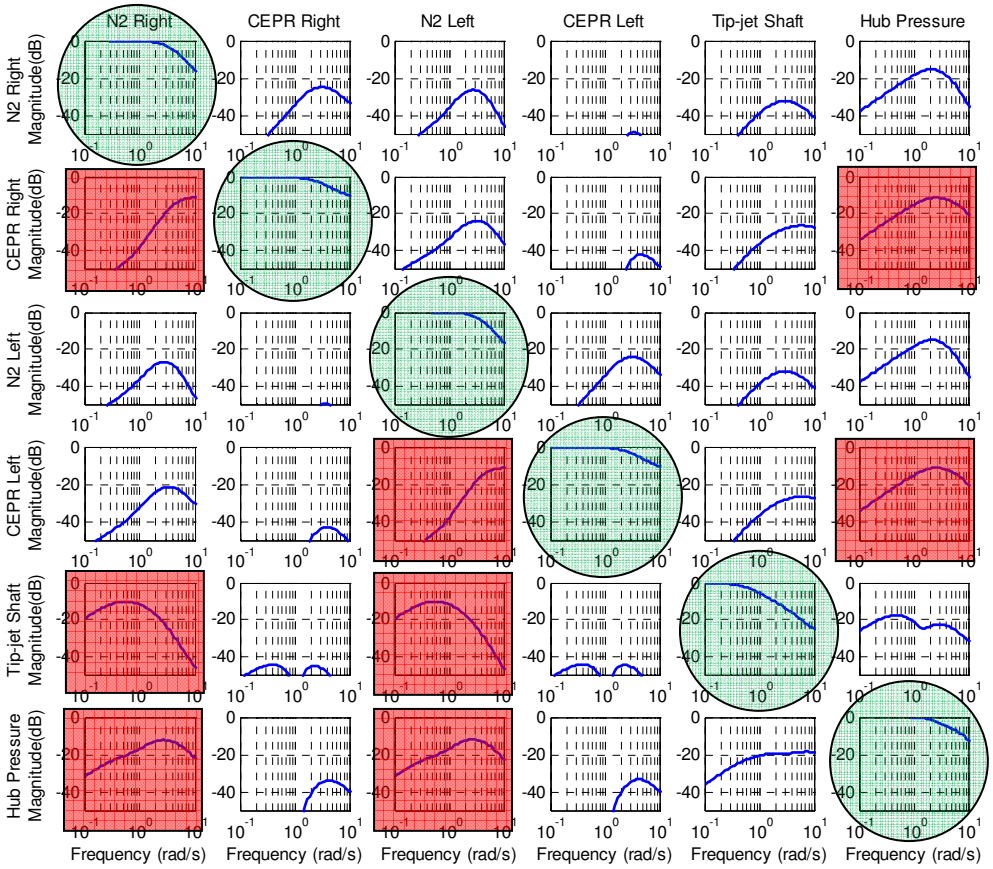

Figure 19: Bode Plot Gains for Baseline PI Controller

Besides bandwidth, phase margin and gain margin are metrics used to gauge control stability performance. A system is considered unstable when the response of the system is close to 180^{o} out of phase with the input signal at frequencies lower than the bandwidth. Although a control system is often designed with the output signal not completely out of phase, uncertainty in gain (magnitude) of the response or delay in the response may result in the control becoming unstable in real life. Gain margin provides an estimate of how close the system is to becoming unstable with respect to uncertainty in the gain; phase margin provides a similar estimate but with respect to uncertainty in the delay of the system. The gain margin is the difference between the magnitude of the response at the bandwidth frequency and the 180^{o} out of phase frequency. Phase margin is the difference between the phase at bandwidth frequency and 180^{o}. Typical values of

gain and phase margin are approximately 2 dB and 30^{o} (SKOGESTAD and POSTLETHWAITE 2005).

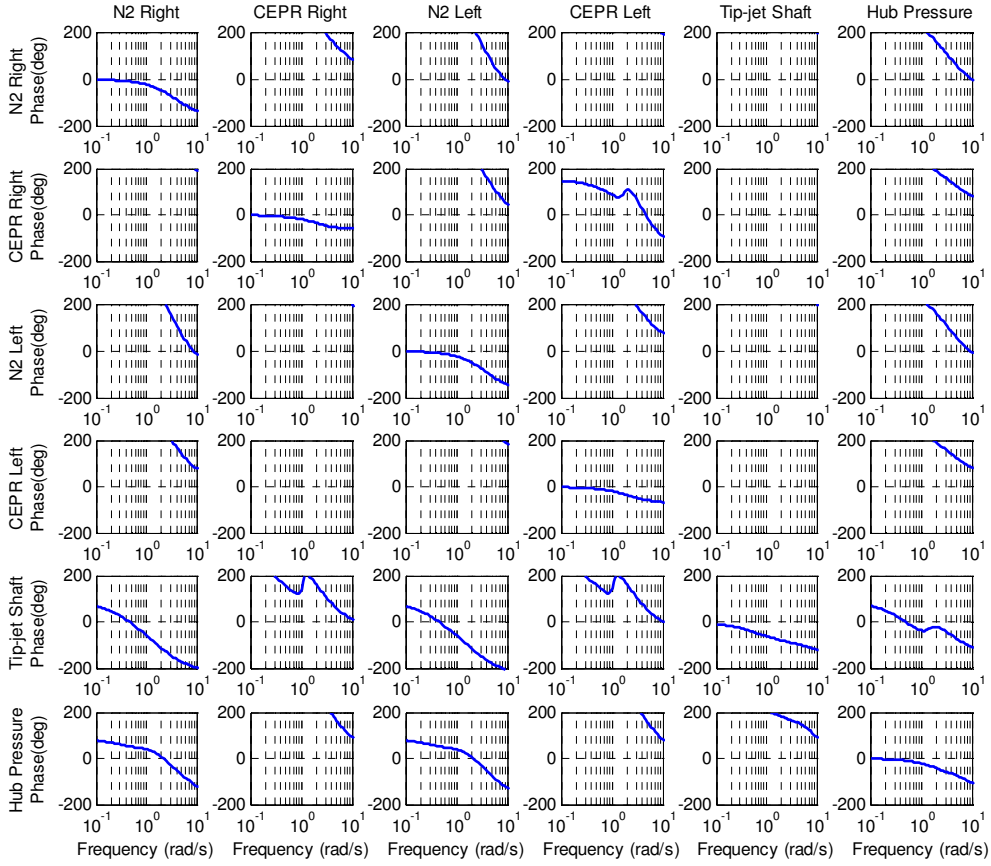

Figure 20: Bode Plot Phase for Baseline PI Controller

Table 12 below summarizes the general performance metrics for the baseline PI controller. After the optimization, the bandwidth of the various control parameters is close to the targets. Additionally both the phase and gain margin for all the control parameters are significantly above the threshold values.

Table 12: Baseline PI Control Performance Metrics

Baseline PI Control	Bandwidth rad/s	Phase Margin degrees	Gain Margin dB
N2 Right	3.13	106	30
CEPR Right	2.72	137	29
N2 Left	3.25	101	28
CEPR Left	2.82	134	32
Tip-jet Shaft	0.64	131	40
Hub Pressure	2.86	124	38

In addition to good control input to output response, it is important that the controlled parameters do not deviate significantly from the demanded values when faced with a disturbance such as a change in rotor load. Figure 21 displays the responses of the system to various different sinusoidal inputs of the rotor load. Besides the tip-jet rotor shaft speed, none of the controlled parameters is significantly affected by a change in rotor load. The maximum response by the tip-jet rotor speed of about -7 dB occurs around 0.6 rad/s. Although not insignificant, the magnitude of the response is less than the -3 dB defined as the criteria for determining control bandwidth. At input frequencies greater then around 3 rad/s tip-jet rotor speed is not affected at all by changes in rotor load. Therefore, the control system should be able to suitably reject changes in the rotor load.

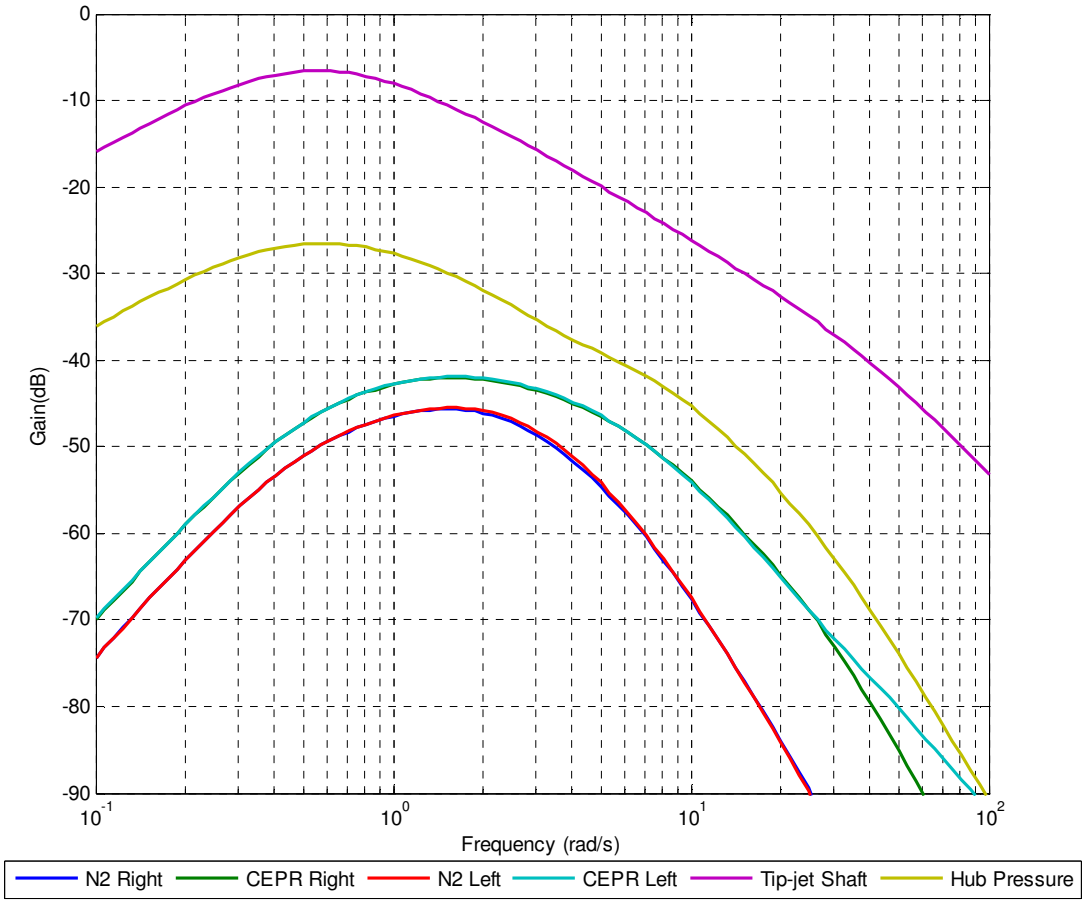

Figure 21: System Response to Rotor Load Disturbance

Based on the analysis of these plots, the control system gains are sized appropriately to meet desired response targets. The next step is to analyze the sized control on the actual tip-jet reaction drive system to analyze the feasibility of the PI controller. If there are significant undesired responses, the PI control may have to either be resized or a different controller may need to be analyzed.

6.4 **Baseline PI Control Simulations**

The purpose of the simulations in the next few sections is to provide an understanding of the overall performance/response of the baseline distributed multivariable PI controller. Specific design variables such as varying throttle settings, disturbance rejection, limit avoidance, and robustness to degradation are explored in

detail. Before any simulations using the PI controller are performed, the PI controller modeling and simulation environment used for these simulations needs to defined

6.4.1 PI Controller Modeling and Simulation Environment

Using the architecture shown in Figure 15 as a guideline, the baseline PI controller modeling and simulation (M&S) environment was developed using Simulink. For this controller architecture, there are two major components that need to be modeled: the tip-jet reaction drive system, G(s), which transposes control inputs to measured output and the PI controller, K(s), which transposes reference tracking errors to control inputs. The tip-jet reaction drive system is represented by the non-linear NPSS defined earlier in section 6.2.1.3. The NPSS model is executed from within Simulink using a custom dynamic link library (DLL) developed by NASA. All the inputs and outputs into this model are normalized around the design point to ensure good matrix properties for the controller design. The PI controller which is designed using the methodology discussed later in section 6.3.1 is modeled using a discrete state-space model. As will be discussed next, the PI controller contains 3 2x2 MIMO controllers that control the fuel flow and exhaust area actuators of each of the tip-jet reaction drive subsystems. The execution of the Simulink based PI controller M&S environment is done from a Matlab command prompt.

6.4.2 Throttle and Rotor Load Changes

The following sections are broken up into analysis of engine throttle changes, rotor load demand changes, and both engine throttle and rotor load demand changes. The engine throttle changes provide an opportunity to analyze how the control responds when multiple demands are changed simultaneously. For a tip-jet reaction drive system, a change in rotor load demand provides a natural opportunity to analyze disturbance

rejection. And the final section will combine the two types of demand changes to see if there are any additional observations.

6.4.2.1 Response to Throttle Change

The first transient to analyze is a change in engine throttle demand. As the engine throttle demand changes, the LP shaft speed and engine CEPR demand also change. The LP shaft speed change is driven primarily to meet a change in thrust demand, whereas the CEPR demand change is defined to maintain a compressor stall margin. Although physically associated with the tip-jet subsystem, the hub pressure demand changes also because of its strong correlation with fan stall margin. The demand schedules were defined and developed while the cycle was operating at new and clean conditions (i.e., component performance matched component map performance).

Figure 22 and Figure 23 below shows a step change in engine throttle demand for a new and clean system. It takes approximately two seconds for the engine shaft speed, CEPR, and hub pressure to settle to the new demand values. Given the 3 rad/s control bandwidth or time constant of 0.3s, this response approximates a first order lag. There is a slight overshoot in CEPR. From the perspective of the tip-jet subsystem, the change in engine throttle has a two-fold effect. Firstly, the tip-jet rotor speed demand remains constant while the demanded hub pressure changes. This is in contrast to the engine subsystem where both speed and CEPR demands change with respect to a throttle change. Secondly the change in engine speed results in a reduction in the mass flow into the tip-jet subsystem which can be viewed as an unmeasured disturbance. Since the amount of torque generated by the tip-jet nozzles is proportional to the amount mass flow, this disturbance has the effect of reducing the tip-jet rotor speed. To overcome this disturbance, the tip-jet fuel flow is increased to generate more torque (as seen through an increase in velocity) in the tip-jet rotor. Since the tip-jet rotor is much heavier relative to the engine, the tip-jet rotor speed takes approximately just over 10 seconds to reject the

load which again approximates a first order lag response for the 2 second time constant of the tip-jet rotor control. The tip-jet rotor deviates from the target approximately 1.5% as a result of this disturbance.

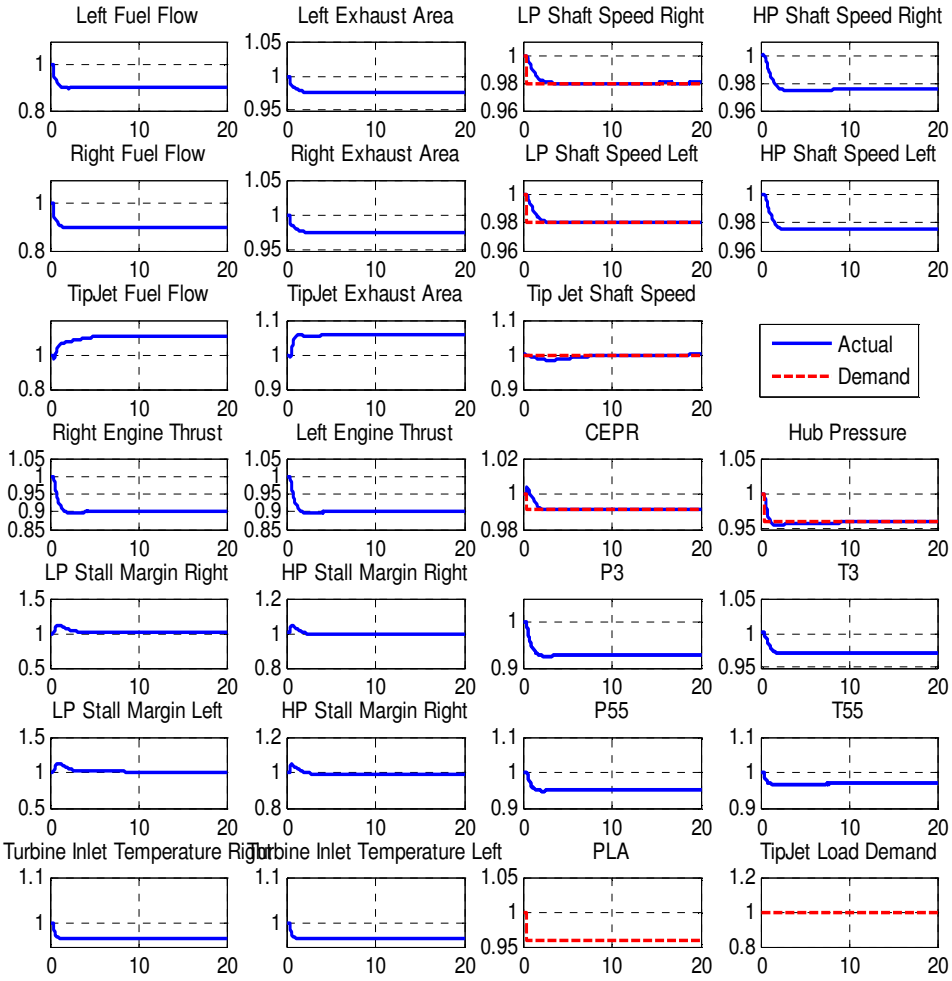

Figure 22: Engine Throttle Demand Decrease for a New and Clean System

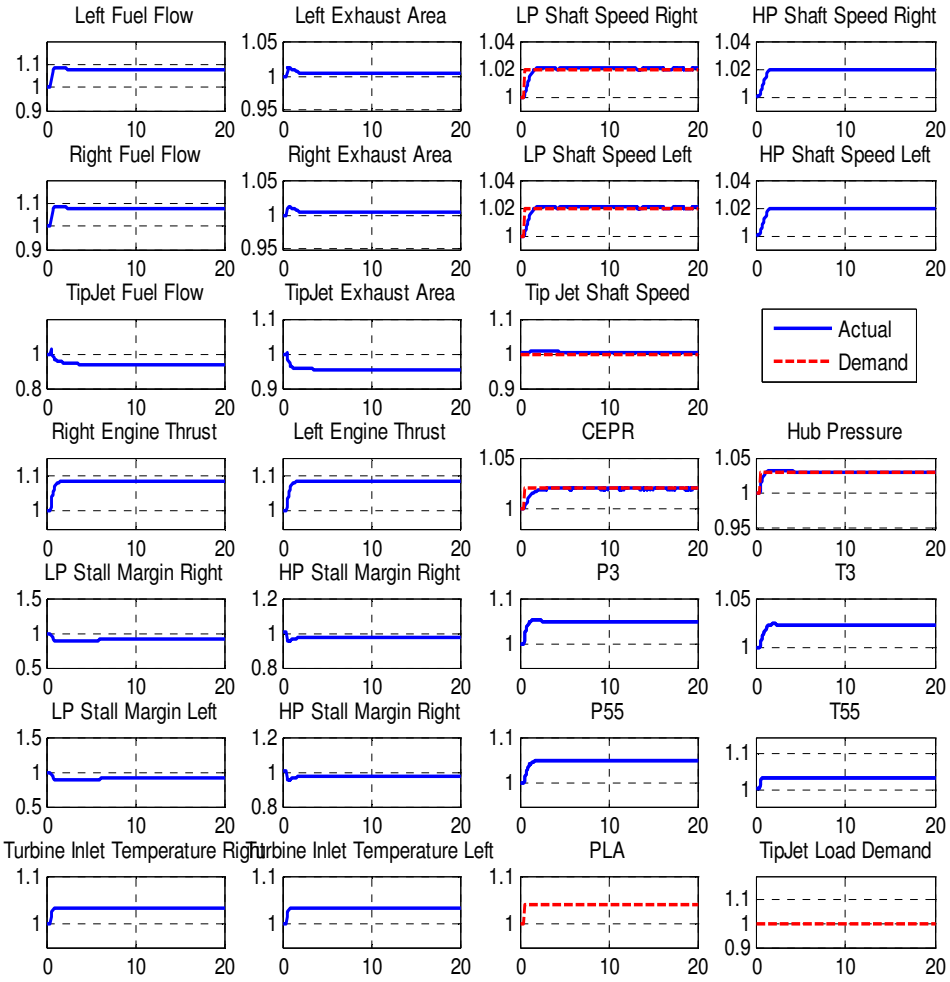

Figure 23: Throttle Demand Increase for a New and Clean System

For parameters such as turbine inlet temperature, which have a significant effect on the residual life of a component (or more importantly the system), an overshoot or more generally deviation from design conditions may result in the margining of the design parameter; this may result in either a much bigger or less fuel efficient system(ADIBHATLA and GASTINEAU, Tracking Filter Selection and Control Model Selection for Model Based Control 1994). The deviation from the design conditions can be seen by looking at plots of an engine throttle change for a degraded system shown in Figure 24 and Figure 25. The values of both measured and unmeasured temperatures

throughout the course of this transient are hotter than their counterparts on a new clean system. For example, turbine inlet temperature increases by almost five percent resulting from degradation. In addition to temperatures the HP shaft speed of the engines operates at higher than design value (design = 1). The response of the control parameters was very similar to the new and clean case with minimal overshoot. However, as the magnitude of the step becomes larger, this may not be the case

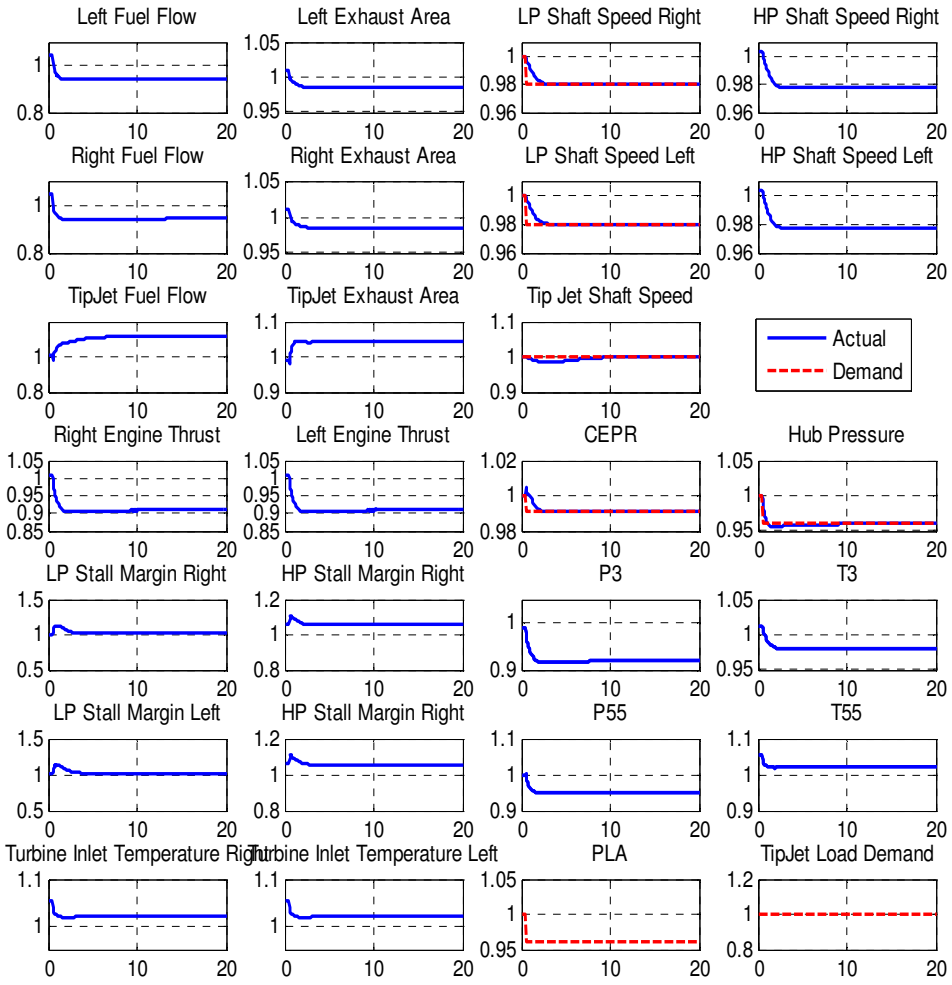

Figure 24: Engine Throttle Demand Decrease for a Degraded System

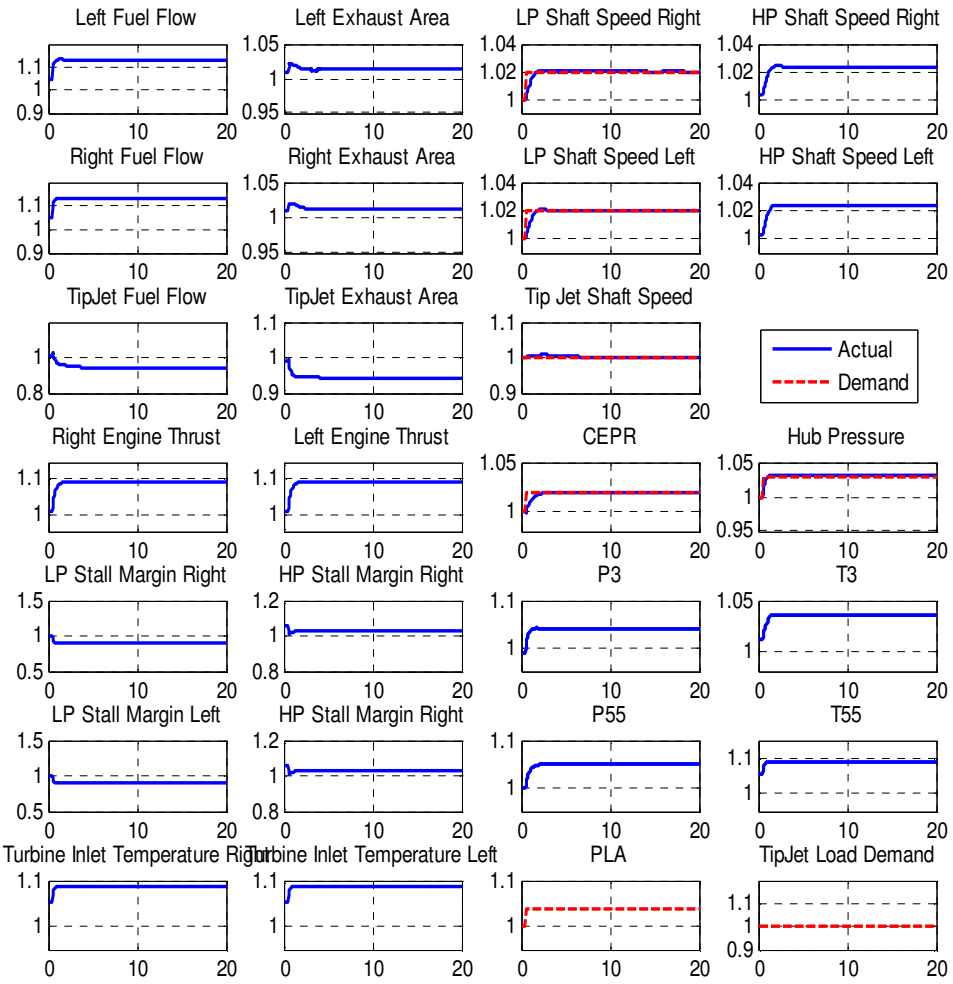

Figure 25: Engine Throttle Demand Increase for a Degraded System

6.4.2.2 Response to Rotor Load Change

Another transient of interest for the tip-jet reaction drive system is the response to a change in tip-jet rotor load. This load change can be induced from either direct pilot input via a change in the collective or indirectly from changes in ambient conditions, such as a wind gust. From the perspective of the tip-jet subsystem control, or more generally the overall control, a rotor load change is viewed as a disturbance. Since it is a

disturbance, none of the speed or pressure demands change when a rotor load change is encountered.

Figure 26, Figure 27, Figure 28, and Figure 29 below show rotor load increase and decreases on both new and clean and degraded systems. Similar to the engine throttle changes, the system response for all four scenarios is very similar. As expected from the disturbance rejection analysis done in the previous section, only tip-jet shaft speed is significantly affected by changes in rotor load. For all the transients, the tip-jet rotor shaft speed takes around 10 seconds to settle after exposed to a 10% change in rotor load. The tip-jet rotor speed deviates approximately 3% from the target. The only significant differences between the new and clean and degraded system cases are the elevated speeds and temperatures of the degraded system. Slight disturbances in the hub pressure can be seen with CEPR and LP shaft speed virtually unaffected. This matches the disturbance rejection analysis performed in the previous chapter.

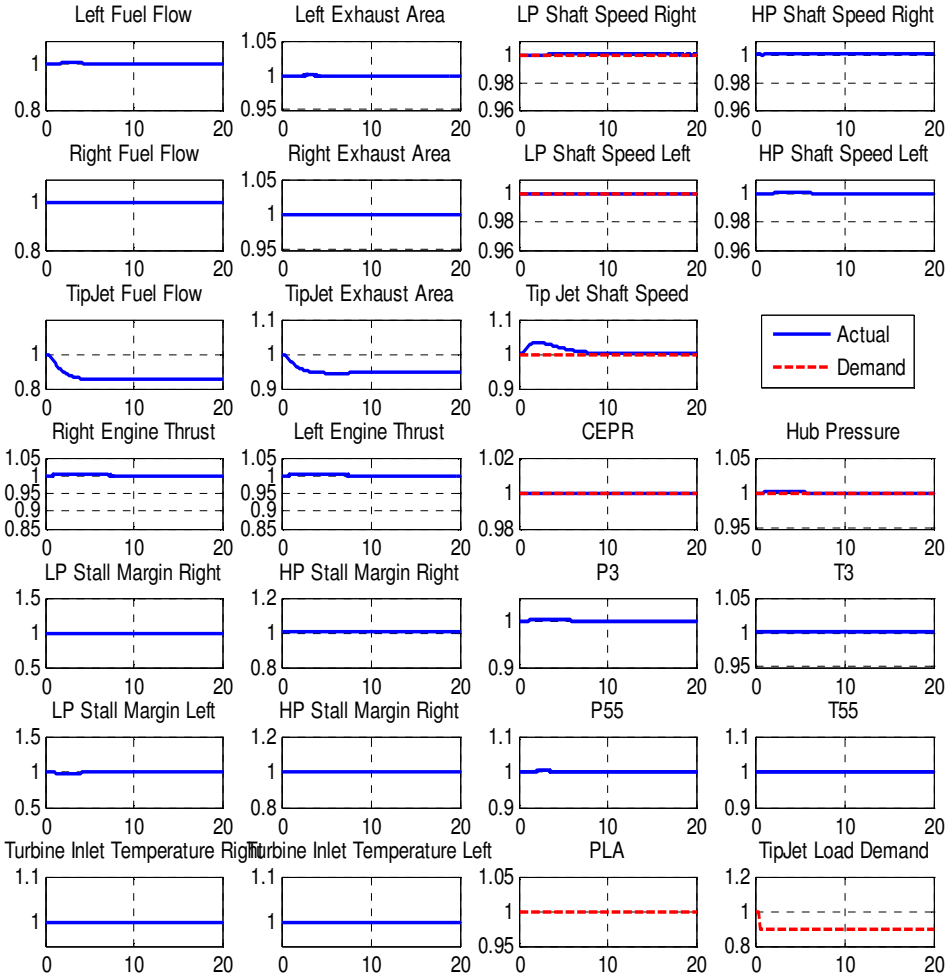

Figure 26: Rotor Load Decrease for New and Clean System

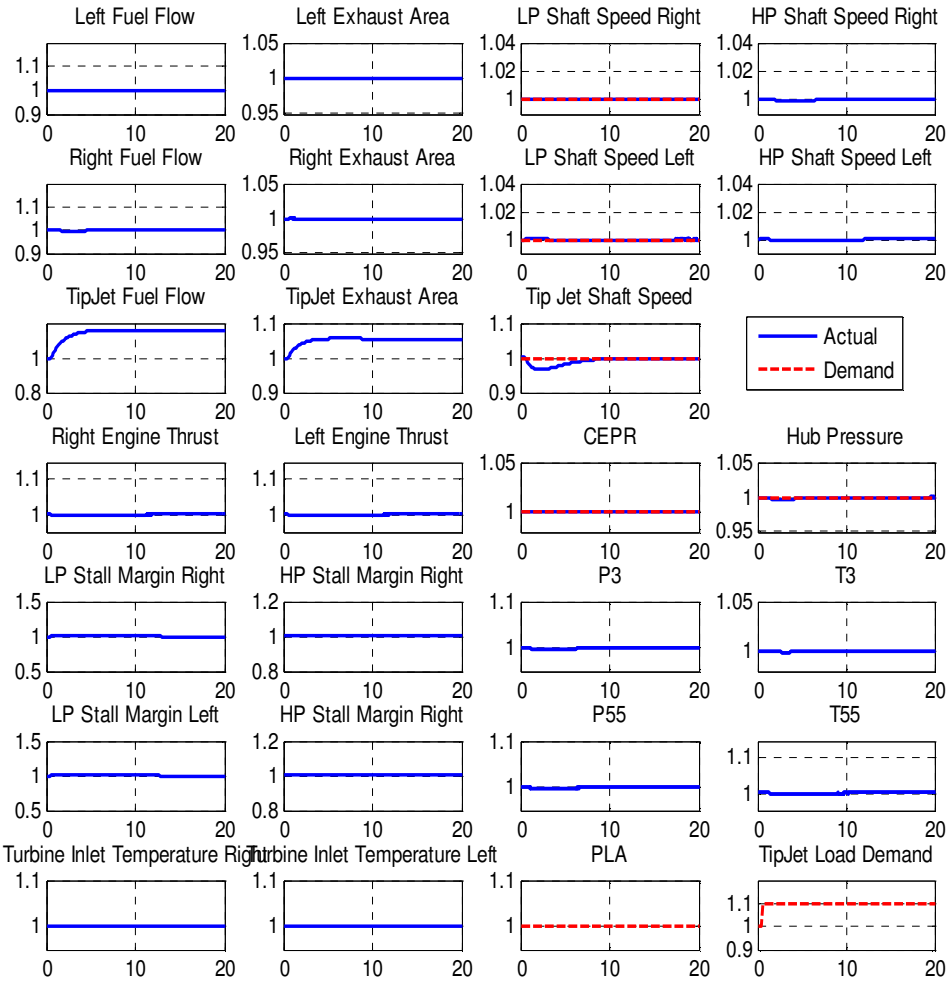

Figure 27: Rotor Load Increase for New and Clean System

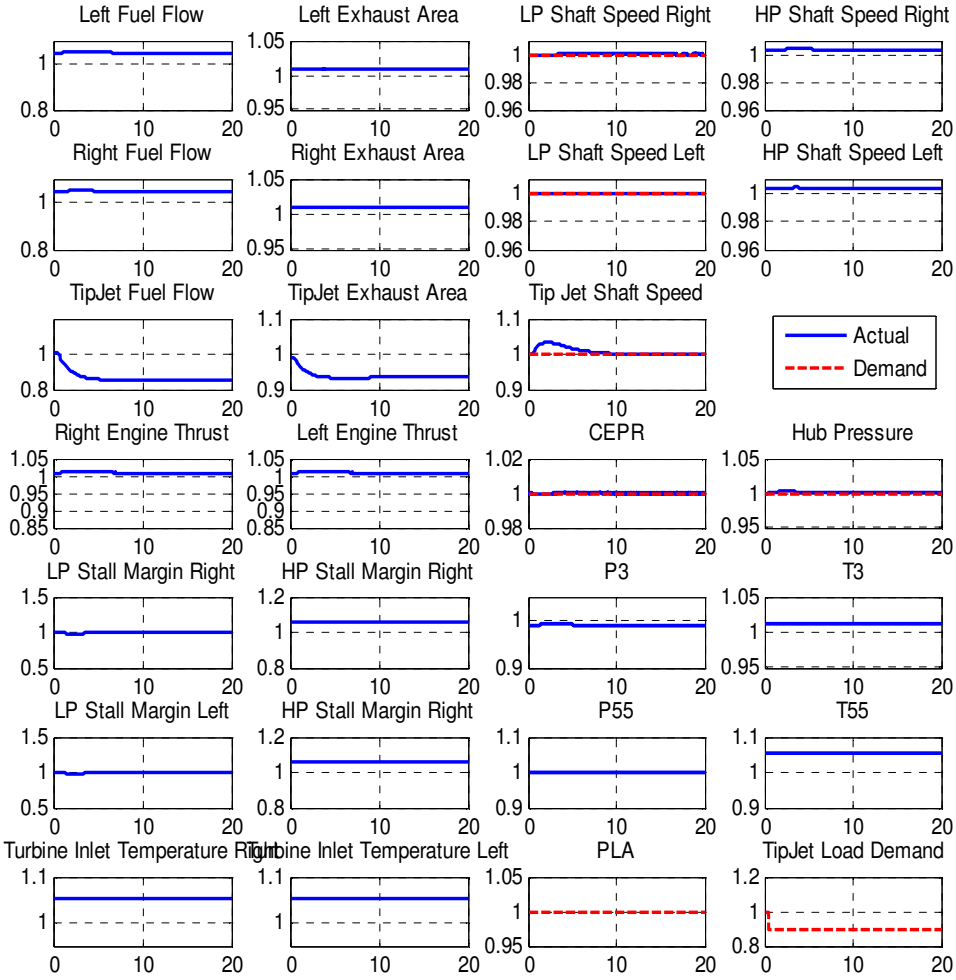

Figure 28: Rotor Load Decrease for Degraded System

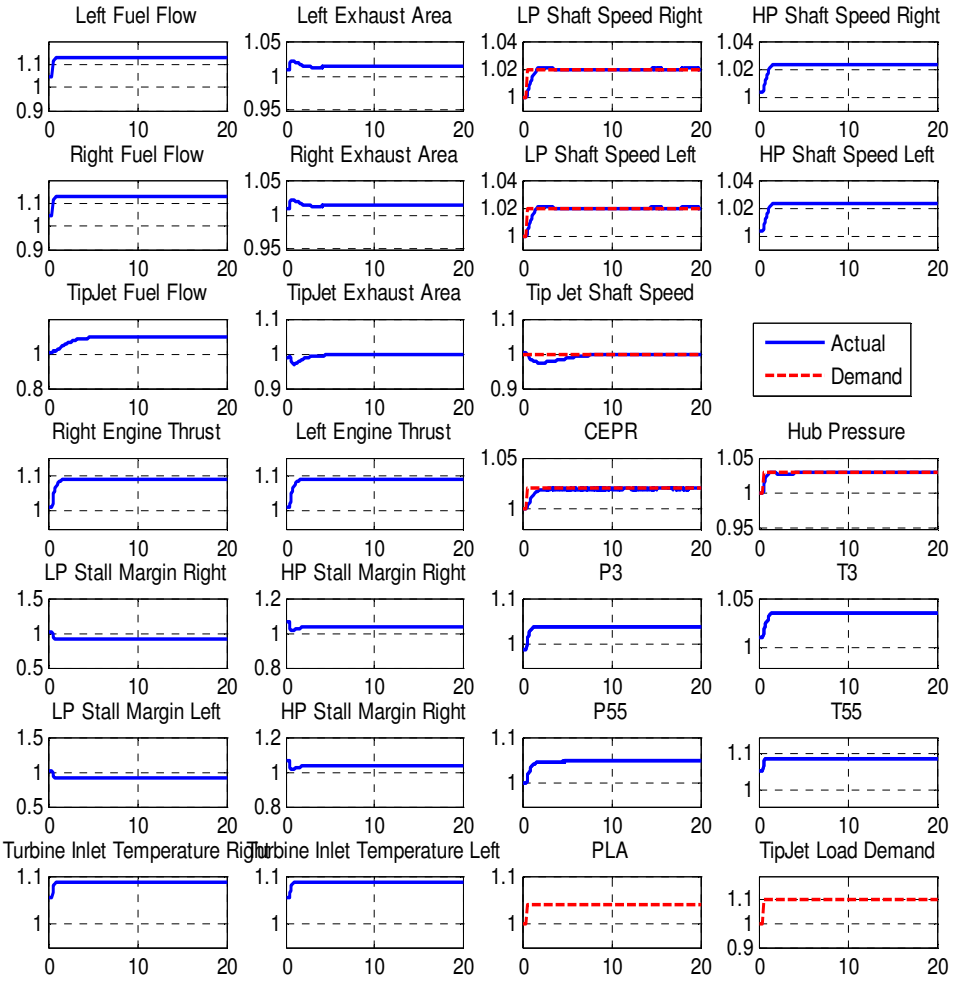

Figure 29: Rotor Load Increase for a Degraded System

6.4.2.3 Response to Throttle and Rotor Load Change

The last transient to analyze combines the two previous transients. Figure 30, Figure 31, Figure 32, and Figure 33 below show rotor load/engine throttle increases and decreases on both new and clean and degraded systems. The tip-jet rotor speed settles approximately 10 seconds after the initialization. And the other demanded parameters settle on the order of two seconds after the transient starts. Both these observations are similar to the observations from the previous runs. There do not appear to be any

additional interactions or effects resulting from the combination of the two demand changes; and the conclusions drawn from the previous sections can be applied to the analysis of these transients.

There is one fairly interesting observation regarding the effect of engine throttle change on the rotor load change response. When the engine subsystem is used in concert with the tip-jet subsystem to reject rotor load demand change, the control is using both mass flow (via the engine demand change) and velocity (via tip-jet fuel flow and exhaust area) to manipulate the system torque. For rotor load demand increases, this scheme offers the benefits of reducing the noise of the tip-jet nozzle exhaust flow, which is proportional to the nozzle velocity. However, the use of changing engine throttle to minimize tip-jet nozzle noise may add additional aircraft handling issues because of the corresponding changes in engine thrust.

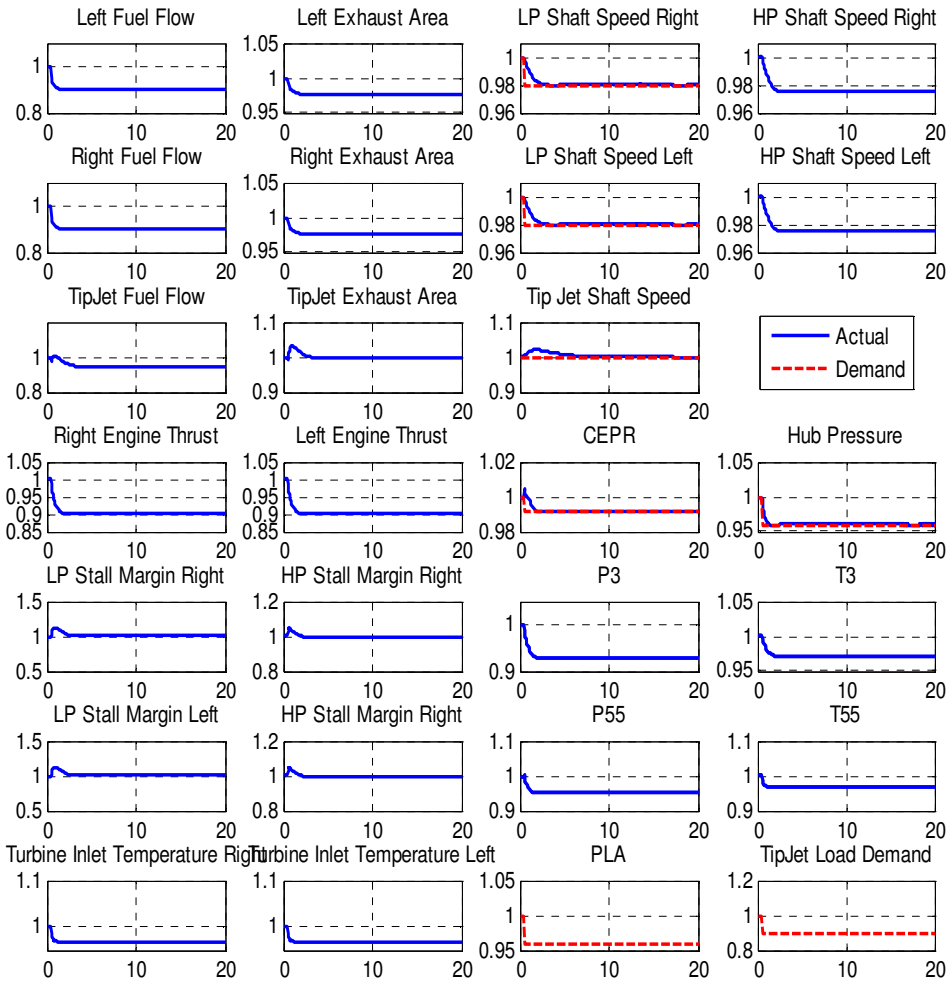

Figure 30: Engine Throttle and Rotor Load Decrease for a New and Clean System

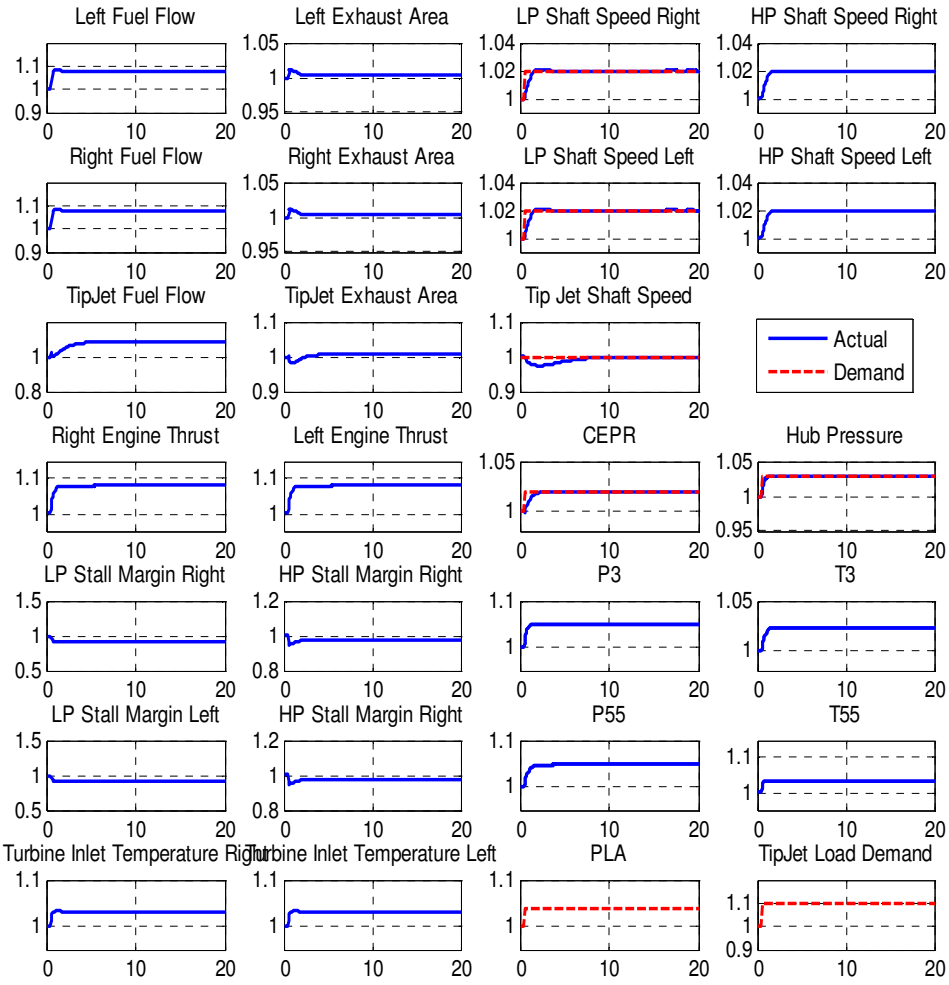

Figure 31: Engine Throttle and Rotor Load Increase for New and Clean System

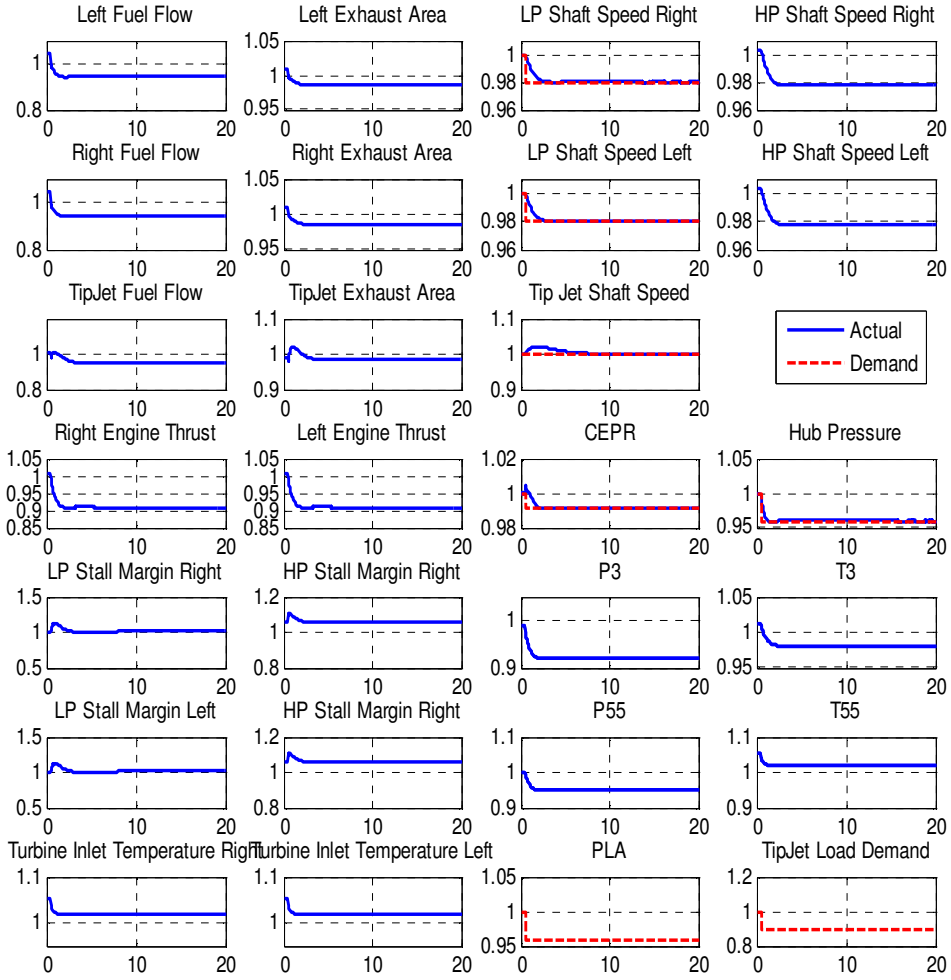

Figure 32: Engine Throttle and Rotor Load Decrease for a Degraded System

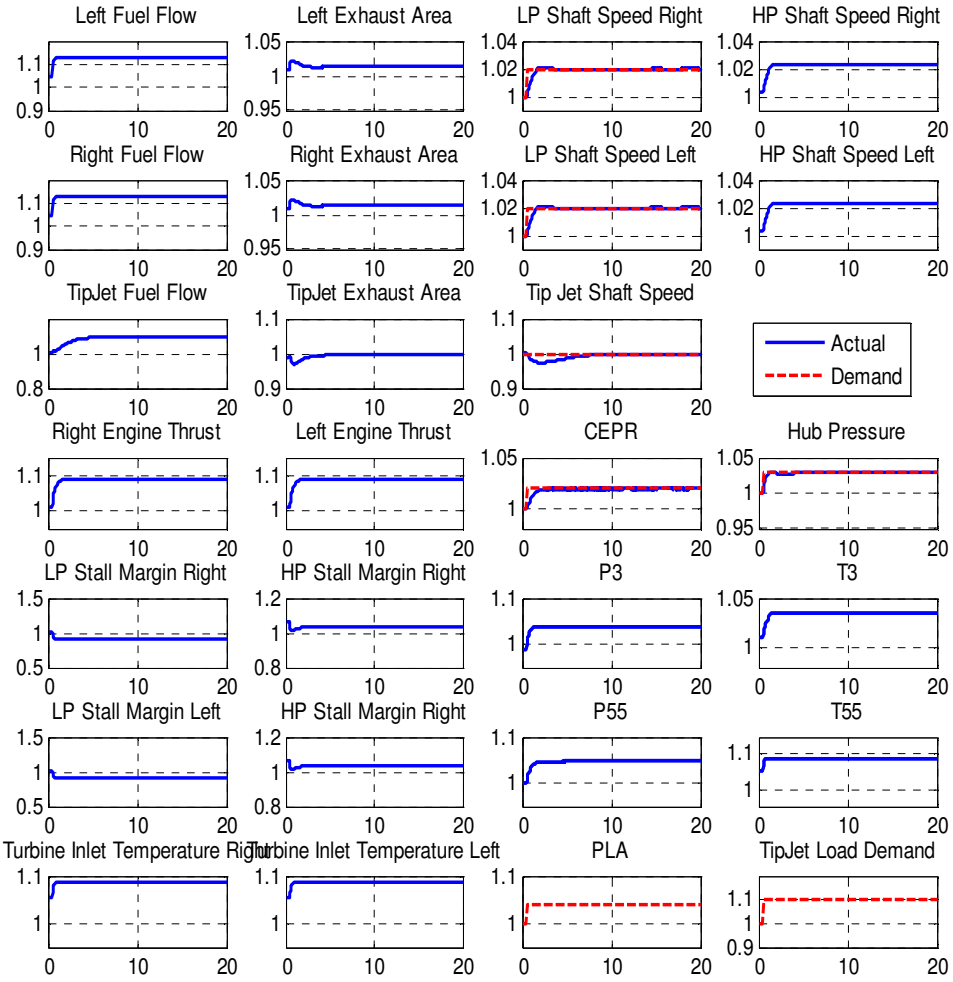

Figure 33: Engine Throttle and Rotor Load Increase for Degraded System

6.4.3 Constraint Handling

Depending on the type of constraint, non-model based control methods like the PI control above use a variety of methods to ensure a constraint is avoided. For certain constraints, such as maximum shaft speed, the value can be measured directly. To ensure these constraints are not violated limits can be placed in the control architecture to adjust the control inputs when a measured value approaches its limit. Other constraints like stall margin or combustor blowout can be correlated to measured parameters. Similar to the

measured parameters, limits can be placed in the control architecture to adjust the control inputs when the limit of a correlated factor is being approached. Other parameters like turbine inlet temperature may not have a strong correlation with measured parameters and thus require margining of the design value to ensure that the parameter does not exceed its limit.

Another set of constraints is defined by the physical limitations of the actuators used on the system. There are limits on the maximum and minimum position of the actuator. Secondly, there is a maximum rate of change of the position of the actuator. Whenever these limits are reached during a transient, the situation is defined as integrator windup. To overcome this limitation an additional windup protection scheme is required (KRISHNAKUMAR, NARAYANSWAMY and GARG 1996).

As the system operates over the flight envelope, it may operate at or exceed the boundary of these constraints. For a PI type control, a separate controller is required for each constraint. Min/max logic would then be used to select the appropriate control action (SPANG and BROWN 1999). Figure 34 below shows an example of how fuel flow is selected based upon a min/max comparison of different controllers.

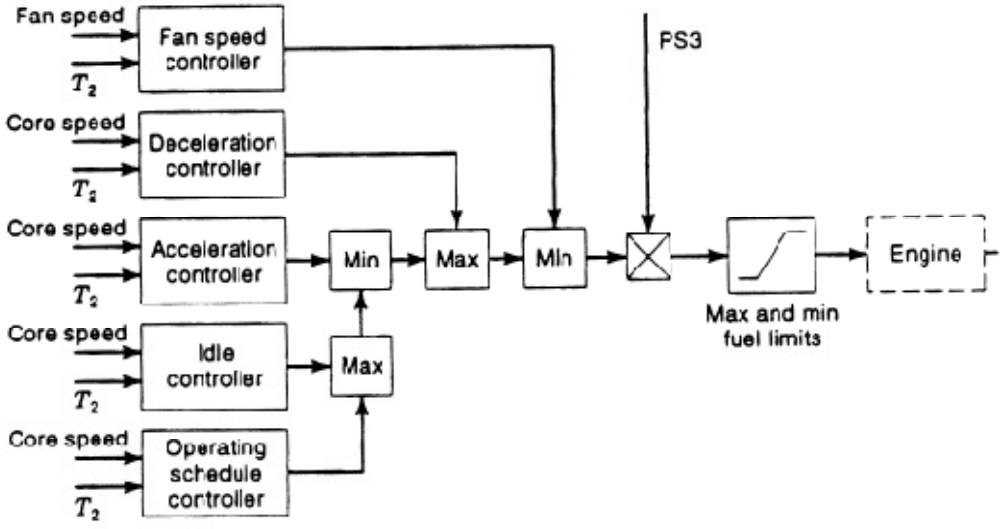

Figure 34: PI Controller Constraint Handling (SPANG and BROWN 1999)

6.4.3.1 PI Controller Fan Stall Margin Handling

To handle stall margin for a PI controller, a separate controller needs to be built. This is a SISO controller where only the control input that most strongly correlates with the constrained parameter will be used to control it. When the constraint limit is approached, the constraint handling controller will override of the demands of the PI controller. This causes the PI controller to operate in a region that has potentially significant suboptimal performance.

Table 13 below summarizes the sensitivity of stall margin to each of the control inputs. The tip-jet exhaust area has the largest sensitivity to with fan stall margin.

Table 13: Stall Margin Sensitivity to Control Inputs

Stall Margin Constraint Relative Gain Array	Right Fuel Flow	Right Exhaust Area	Left Fuel Flow	Left Exhaust Area	Tip-Jet Fuel Flow	Tip-Jet Exhaust Area
Fan Stall Right	0.0822	0.2651	0.0872	0.0706	0.0455	0.4493
HPC Stall Right	0.444	0.5283	0.0147	0.011	0.0022	-0.0003
Fan Stall Left	0.0873	0.0705	0.0824	0.265	0.0455	0.4493
HPC Stall Left	0.0146	0.0112	0.4434	0.5288	0.002	-0.0001

However since the PI controller is non-model based, a measured parameter that correlates with stall margin is needed to define when a stall margin limit is being approached. Assuming the flow out of the tip-jet nozzle is choked a flow function calculation can be used to approximate operating line excursions towards a stall margin limit. If the flow is choked the gas properties are the, a flow function can be assumed to be a constant function of mass flow, w, temperature, T, pressure, P, and area, A and can be written as follows.

$$\frac{w\sqrt{T}}{PA} = const. \qquad [124]$$

Where mass flow is proportional to fan speed, N, and temperature can be approximated by a ratio of fuel flow, w_f, and fan speed. The flow function equation can then be rewritten as follows.

$$\frac{\sqrt{Nw_f}}{PA} = const. \qquad [125]$$

For a given numerator, as pressure increases, the area must decrease to maintain a constant flow function. As pressure increases, the fan stall margin decreases and there must be a certain value of the area which represents the stall margin limit. Therefore this value becomes the lower limit of the tip-jet nozzle exhaust area.

To demonstrate PI controller stall handing capability a simulations were run where the hub pressure demand increased while holding all other demands constants. In this case the fan pressure ratio will increase resulting in the system operating at a lower stall margin steady-state fan stall margin. Figure 35 and Figure 36 below shows plots of both the unconstrained and constrained simulations. The former is unconstrained and the latter has a constraint on the LP stall margin. For the purpose of these simulations, the stall margin was constrained to be no lower than 75% of the design value. In reality this value would be much lower, however to ensure the nonlinear system model converges in the unconstrained case, the stall margin limit was defined at such a high level. In the constrained simulation, the LP stall margin constraint becomes active resulting in the system operating with a lower than demanded hub pressure.

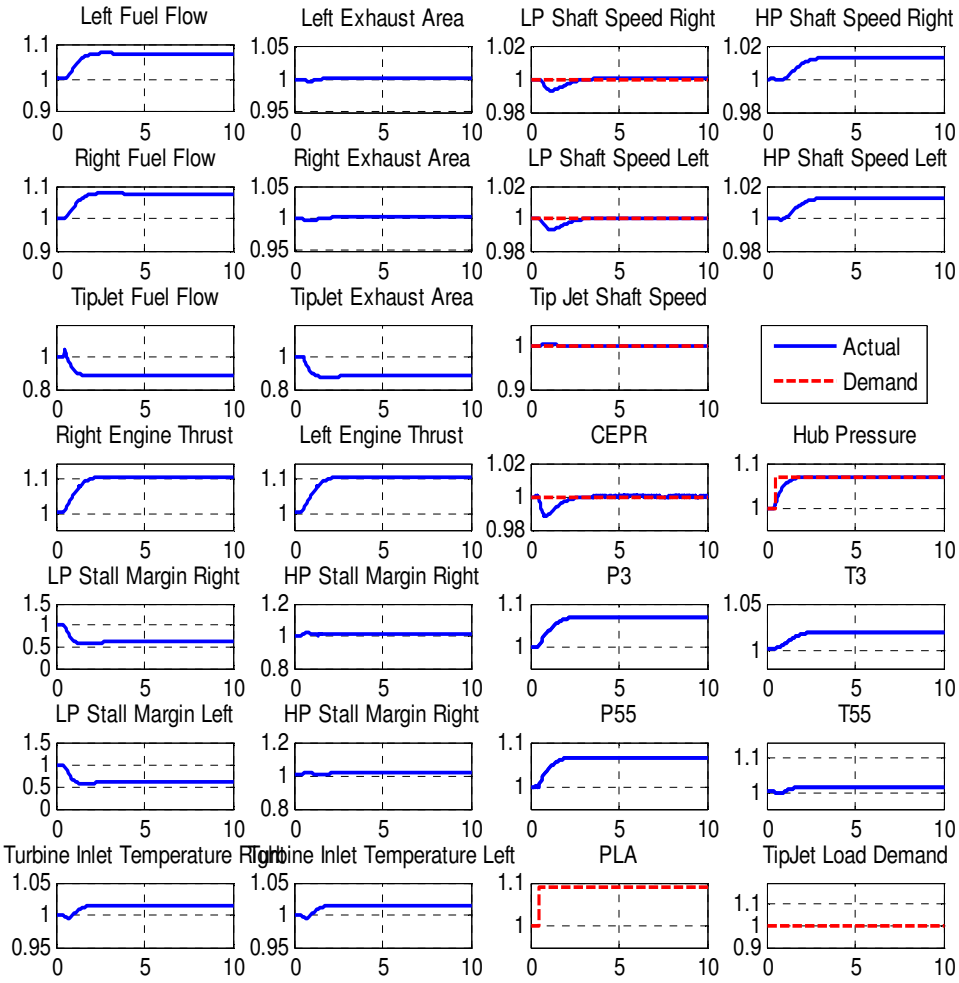

Figure 35: Unconstrained Hub Pressure Increase

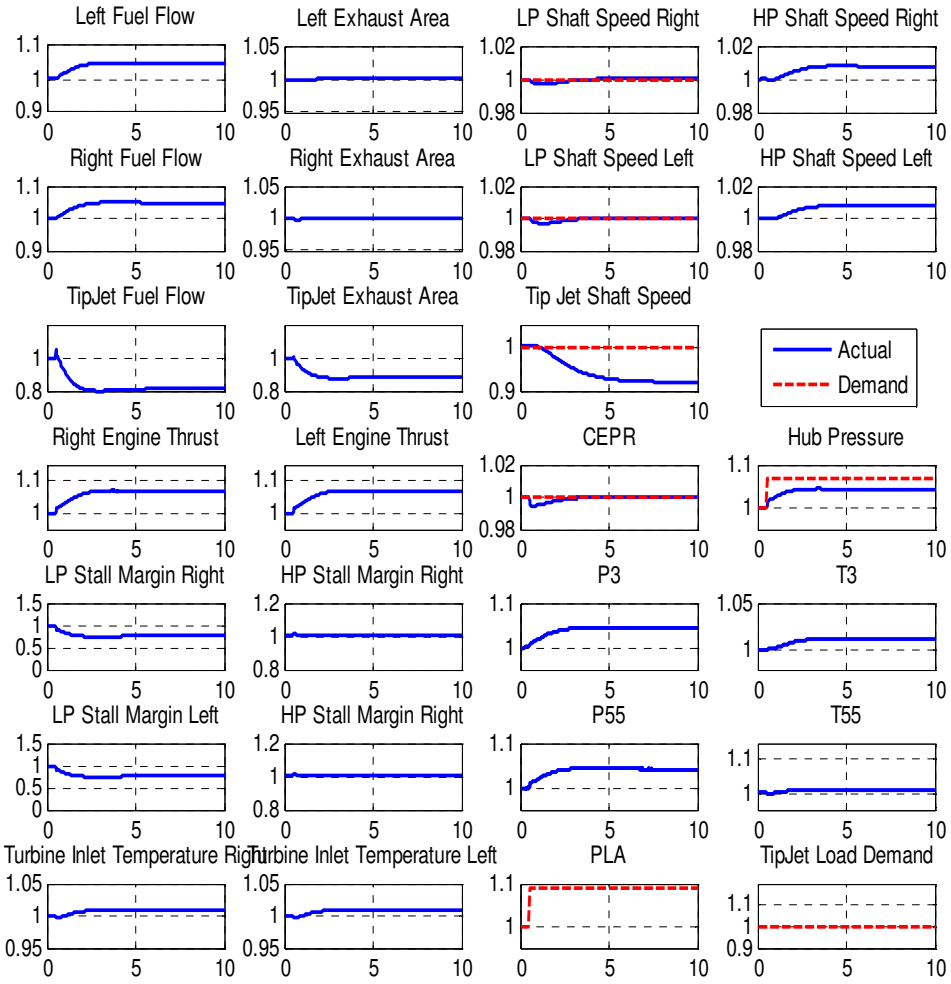

Figure 36: Stall Margin Constrained Hub Pressure Increase

In addition to LP stall margin, there are significant differences between the unconstrained and constrained tip-jet shaft speed, CEPR, and hub pressure. Figure 37 below shows these differences in detail. Most noticeable is the tip-jet shaft speed operating at steady-state speed almost 10% lower than the target value when the stall margin constraint is hit. This condition may result is significant aerodynamic issues with the tip-jet rotor blades and result in quite undesirable vehicle handling properties. As will

be seen in the next few chapters, MPC offers significant improvements in the control/system performance when constraints stall margin constraints are encountered and that capability is one its primary selling points..

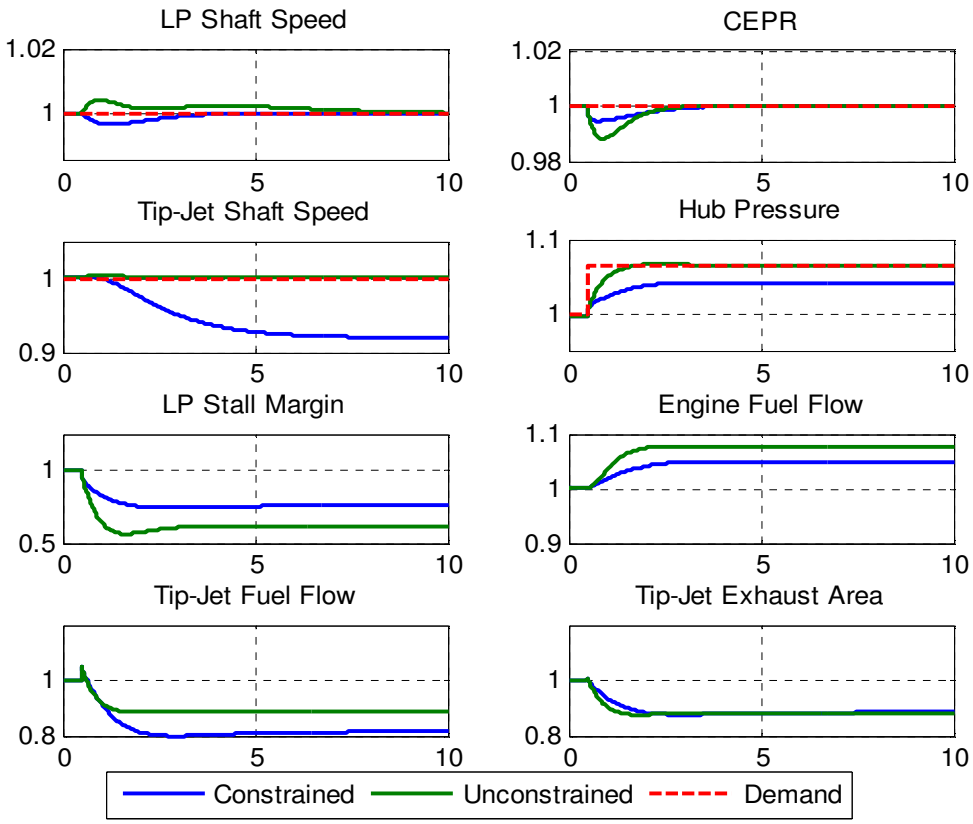

Figure 37: Baseline PI Control Performance for Stall Margin Limit Avoidance

6.5 Baseline PI Control Summary

In this chapter, the baseline multivariable PI controller for the tip-jet reaction drive system was developed. Before the controller was sized, the significance of dynamics present of a tip-jet reaction drive system was defined. It was determined that unless the duct volumes were significantly larger than a component volume, the gas path dynamics can be neglected. Given the understanding of the dynamics and the available sensors on the system, the inputs and outputs of the controller was defined. It was determined that three separate 2x2 controllers would be appropriate. Once sized, the

frequency properties of the controller were analyzed to ensure the performance of the control was acceptable. After this, different transient simulations were run that captured the response of the system varying throttle settings and changing rotor load. Stall margin constraint handling was also demonstrated with a noticeable degradation in controller performance. Given this baseline PI controller, a centralized MPC can be developed. This will provide a baseline for predictive controller performance.

CHAPTER 7

CENTRALIZED MODEL PREDICTIVE CONTROL

In addition to the baseline PI controller sized and analyzed in the previous chapter, a centralized MPC will be sized and analyzed for the purposes of comparison with the proposed MPC architecture. Although the computational burden for a centralized MPC makes it infeasible, it will provide the gold standard for MPC performance on the tip-jet reaction drive system.

7.1 **Centralized MPC Design Process**

The sizing and analyzing of the centralized MPC is a multi staged process. Figure 38 below shows a general step-by-step process that was followed to create the baseline centralized MPC controller. As with the PI controller design process, the centralized MPC design process can be broken up into three general categories: controller architecture components definition and modeling, controller design and sizing, and transient simulations. Within each of these general categories are multiple tasks.

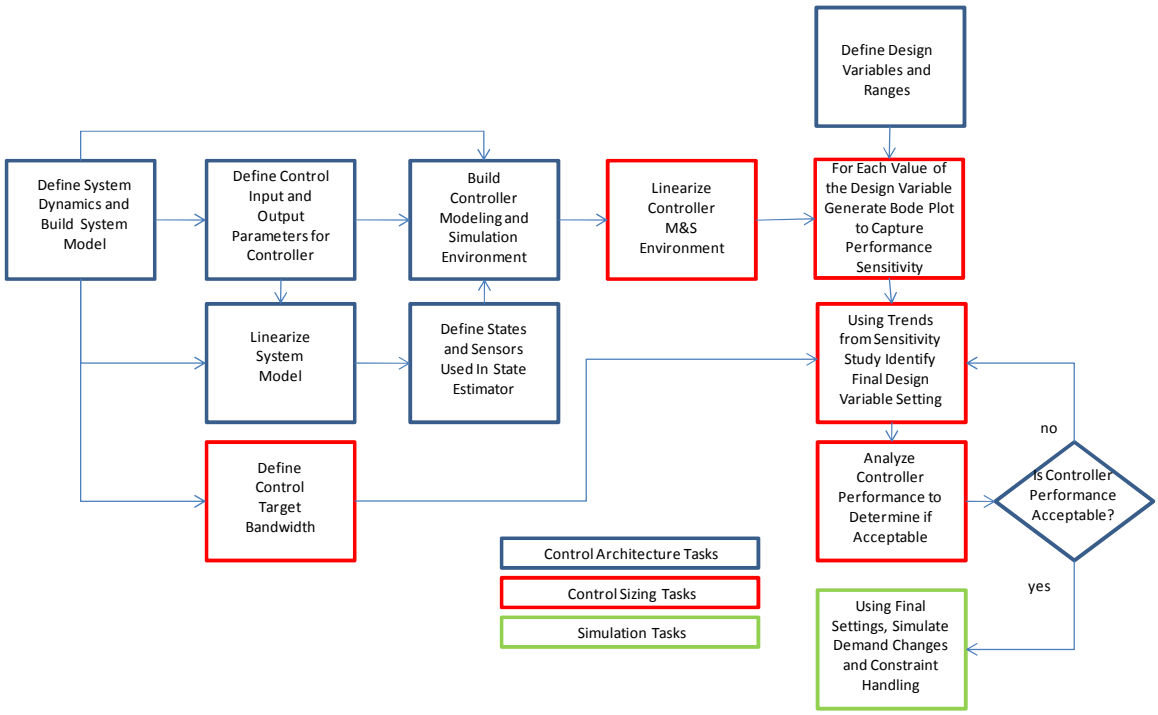

Figure 38: Centralized MPC Design Process

The first controller design category has six tasks. The first task is to define and quantify the dynamics of the system and build a non-linear system model of the tip-jet reaction drive system. This task is described earlier in detail in section 6.2.1. The second task is to linearize the tip-jet reaction drive system model. The linear tip-jet model is used both in the state estimator and in the centralized MPC algorithm. The third task is to define the appropriate control input and output parameter. As with the PI controller, this task uses both the non-linear and linear models. The details of this task are discussed in detail in section 7.2.1. The fourth task, which is described in section 7.2.2, is to define which states and sensors are to be used for the state estimator. After designing the state estimator, the various elements of the centralized MPC as well as the relevant design variables and ranges can be defined. This discussion is covered in section 7.2.3. As opposed to the PI controller, where the PI controller M&S environment was used in the controller sizing, the centralized MPC needs to have the M&S environment defined

before the controller can be sized. The details of the centralized MPC M&S environment are discussed in section 7.2.4. Once these tasks are completed the controller can be sized.

Unlike the PI controller, which used an optimization algorithm to size the controller, the centralized MPC sizing process involves a sensitivity study to size the controller. The first task in the controller sizing category is to define the target bandwidth of the controller. For this dissertation, the target bandwidth was defined using the system dynamics outlined in the initial step in the control design process. The second task is to linearize the controller M&S environment. Once the control environment is linearized, the sensitivity can be performed fairly quickly. The sensitivity study is performed by varying each design variable across its range and then generating a Bode plot for each design variable setting. Using the trends in the Bode plot, a final design variable selection can be made. The details of these two tasks are covered in section 7.3. Once the controller is sized, the last task is to analyze the performance of the controller. The details of this task are discussed in sections 7.3.1.4, 7.3.2, and 7.3.3. If the controller does not have acceptable performance properties, the sensitivity study results can be revisited to see if there is another suitable option.

Once a controller with acceptable performance qualities has been sized, various transient simulations can be analyzed. Using the M&S environment, simulations varying both throttle demand and rotor load changes can be run. The results of the simulations and discussion of the results is contained in section 7.4.1. The last simulation task to be performed is demonstration of PI controller constraint handling. The details of the constraint handling simulation are discussed in section 7.4.2. Once all these tasks are complete, the performance capabilities of a centralized MPC are understood and comparisons with other controllers can be made.

7.2 **Centralized Model Predictive Control Development**

Figure 39 below shows centralized MPC architecture in detail. There are some obvious differences between a centralized MPC architecture and that of the PI controller architecture in Figure 15. First and foremost is a completely different controller. In addition to a different controller, there are two additional components, the onboard model (G_{ob}) and the state estimator. The onboard model is used to estimate unmeasured parameters such as thrust which will be used in the controller. The state estimator compares the actual system output to the output of the onboard model and adjusts the estimate of the onboard model state so that the two match. The state estimator is required when all the states of the system cannot be physically measured and to ensure that the onboard model matches the measured output of the system. Although not explicitly shown on the figure, the onboard model is used in the controller.

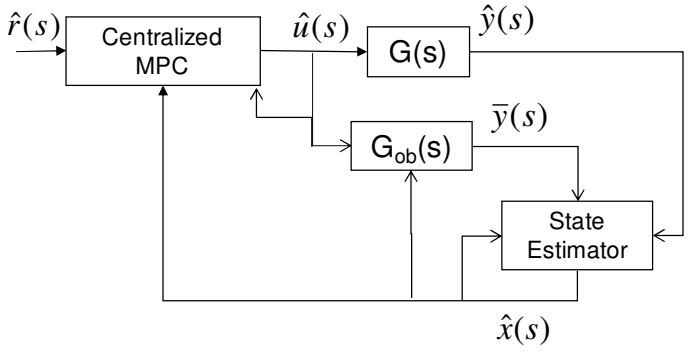

Figure 39: Centralized MPC Architecture

Before the MPC controller is designed a few aspects of the MPC architecture need to be developed. Now that a state estimator has been added to the system, unmeasured parameters can be estimated. Therefore, there is potential for different parameters to be controlled. As in the previous chapter, a relative gain array along with a probabilistic analysis of the system performance will be used to determine the appropriate input/output combinations. Once these inputs and outputs have been defined, it is necessary to design the state estimator. In doing so, the necessary states and measurements will also be defined. Given the dynamics of the systems which were

defined in the previous chapter and the controller input/output selections, the MPC algorithm that will be used in the controller can be defined.

7.2.1 Model Based Control Input/Output Selection

The steady state design of the non-model based PI controller can be a starting point for steady state design of a model based controller. In addition to shaft speeds and pressure ratios, other parameters such as thrust and stall margin can be controlled. As with non model based control there are a couple more general metrics that must be taken into account when selecting which variables are to be controlled: 1.) minimal interactions between inputs and outputs and 2.) minimal variance of uncontrolled parameters of interest over the functional life of the system.

7.2.1.1 Relative Gain Array

Table 14 below shows the steady state relative gain array (RGA) comparing different potential input/output combinations of both measured and unmeasured parameters. (RGA tables generated at other frequencies are located in Appendix C) A value of 1 in the table represents a significant relationship between input and output and the converse holds true for a value of zero. To minimize interactions, the choice of the final input/output combination should be as diagonally dominant as possible (i.e., the input/output value should be as close to 1 as possible).

Table 14: Relative Gain Array of Tip-Jet Reaction Drive System Model Based Control at 0 rad/s

Relative Gain Array	Right Fuel Flow	Exhaust Area	Left Fuel Flow	Exhaust Area	Tip-Jet Fuel Flow	Exhaust Area
Thrust Right	0.5506	0.0248	0.0124	0.0125	-0.0487	0.0683
LP Shaft Right	0.0539	0.0482	-0.004	-0.0041	0.0177	-0.0218
HP Shaft Right	0.0517	0.0042	-0.0014	-0.0014	0.006	-0.0075
Fan Stall Right	0.2063	0.1775	0.0596	0.0605	-0.0808	0.5368
HPC Stall Right	0.0393	0.0509	0.0025	0.0025	-0.0078	0.0119
CEPR Right	-0.0034	0.5913	0.0107	0.0109	-0.0465	0.0588
Thrust Right	0.0123	0.0125	0.5508	0.0248	-0.0487	0.0683
LP Shaft Left	-0.004	-0.0041	0.054	0.0482	0.0177	-0.0218
HP Shaft Left	-0.0014	-0.0014	0.0517	0.0042	0.006	-0.0075
Fan Stall Left	0.0597	0.0605	0.2063	0.1775	-0.0808	0.5367
HPC Stall Left	0.0025	0.0026	0.039	0.051	-0.0081	0.0122
CEPR Left	0.0108	0.011	-0.0034	0.5914	-0.0466	0.0588
TipJetRotorShaft	-0.0004	-0.0006	-0.0004	-0.0007	1.3867	-0.3847
Hub Pressure	0.0222	0.0226	0.0222	0.0227	-0.066	0.0915

From Table 14, the outputs that are most affected by the inputs are engine thrust, CEPR, tip-jet shaft speed, and fan stall margin. A review of fan stall margin as an option identified some significant issues for several reasons. First due to the symmetry of the system, including fan stall margin will result in seven outputs that will need to be controlled by six inputs. This makes the control problem underspecified which may lead to numerical issues when trying to minimize the cost function in the MPC algorithm. Additionally, fan stall margin is affected by all the control inputs. Care would have to be taken to ensure the interactions were minimal. These two observations led to the selection of hub pressure as a control output. Although there is less of a functional relationship between the inputs and pressure relative to stall margin, the knowledge of pressure has less interactions and its inclusion would make the matrices used in the MPC cost function square. The next control parameter selected was tip-jet shaft speed because of its strong functional relationship with tip-jet fuel flow and tip-jet exhaust area. Thrust was a natural choice for inclusion in the controller due to its large functional relationship to engine fuel, as shown in Table 10. Although CEPR is not a model estimated parameter, the functional relationship between it and engine exhaust area is

much larger than HP stall margin. Therefore CEPR was chosen to be a control parameter for the centralized MPC. However, minimum levels of both LP and HP stall margin will be integrated into the MPC controller as constraints. With 6 control outputs selected to correlate with the predetermined inputs, a final check of steady state performance variation is necessary.

7.2.1.2 Steady State Performance Variation

Similar to the non model based control, before any final choices on control input/output combinations are made, the variance of the uncontrolled performance metrics over the life of the system need to be analyzed. The input/output control pairings should be chosen such that variance of unmeasured parameters is minimal over the useable life of the system.

Table 15 below compares variance of a model based control with hub pressure, tip-jet shaft speed, engine thrust, and CEPR as controlled parameters to the non-model based control from last chapter for both a new and clean system and a degraded system. The model used in the model based controller is assumed to have no errors. Comparing the mean values of a new and clean system versus a degraded system captures the effects of degradation. The standard deviation in parameters captures the effect of machine to machine variation. By comparing the mean and standard deviation values between model based and non model based control, it is possible to show the effect of the different types of controls.

Table 15: Model Based and Non-model Based Control Performance Variance Comparison over Useful Life of System

		Non Model Based		Model Based	
Parameter	Age	Mean	Standard Deviation	Mean	Standard Deviation
Engine Thrust	New	0.999	0.008	1.000	0.000
TSFC	New	1.004	0.014	1.004	0.018
LP Shaft Speed	New	1.000	0.000	1.003	0.010
HP Shaft Speed	New	1.000	0.013	1.000	0.010
Turbine Inlet Temperature	New	1.004	0.012	1.004	0.015
LP Stall Margin	New	1.000	0.008	0.997	0.061
HP Stall Margin	New	0.983	0.076	0.987	0.077
Engine Thrust	Degraded	1.008	0.011	1.000	0.000
TSFC	Degraded	1.040	0.014	1.032	0.017
LP Shaft Speed	Degraded	1.000	0.000	0.995	0.008
HP Shaft Speed	Degraded	1.002	0.013	0.999	0.010
Turbine Inlet Temperature	Degraded	1.041	0.011	1.033	0.014
LP Stall Margin	Degraded	1.002	0.007	0.939	0.074
HP Stall Margin	Degraded	1.049	0.075	1.059	0.080

Model based control has some significant benefits relative to non model based control. For non model based control, as the system ages, the thrust increases by approximately 1%. Additionally there is an approximate 1% variation in thrust resulting from machine-to-machine variation. Assuming minimal modeling errors, there is virtually no thrust variation of the life of the system. However, the incorporation of modeling error, may affect the variation of thrust over the life of the system. This factor will be explored later in this chapter. The increase in turbine inlet temperature (and temperatures in general) over the life of a system with a model based control is approximately 1% less than a system with a non model based control. This factor may reduce the amount of steady state margining required in selecting the design turbine inlet temperature. Additionally, as the system ages with model based control, the TSFC degrades by about 1% less relative to non model based control.

However, there is the one main drawback to using model based control: the approximate 7% increase in fan stall margin variation. The main reason this happens is because the fan speed varies while the hub pressure remains constant. This factor may

result in the choice of a slightly lower design fan pressure ratio, which has an adverse effect on the system performance. However, for MPC, minimum stall margin can be included as a constraint to ensure safe operation at all times and may not require any margining.

From this analysis, it was determined that the use of model based control would allow the inclusion of engine thrust as a control parameter. Although stall margin is a potential control option for model based control, CEPR and tip-jet EPR were chosen because of minimal interactions and improved matrix properties for solving the MPC cost function. Given these selections, the components of the MPC control algorithm can be defined.

7.2.2 Onboard Model and State Estimator Performance

The purpose of the onboard model is to estimate unmeasured system performance parameters such as thrust and stall margin. These estimates will then be used by the controller to determine the appropriate control action. The onboard model may not match the actual performance of the system because of simplifications made to the model, inherent errors in the model, or a change in performance of the system over time as the result of degradation or a fault.

Although potentially negative in terms of model accuracy, model simplifications, such as linearization of the model, or use of a 0-D modeling technique, can allow certain design and simulation capabilities to be incorporated and can even make the models run faster. Model simplification is incorporated twice in a MPC. Firstly, a linear approximation of the model is used to significantly reduce the computational burden of the MPC. Secondly, a linear approximation is often used in the state estimator, which allows for the use of linear state estimation techniques such as the Kalman Filter (LUPPOLD, et al. 1989).

Although modeling error is functionally the same as model simplification, it is categorized differently. Here modeling error is defined as the incorrect modeling of the physics of the system. A couple of examples of modeling error on the tip-jet reaction drive system are incorrect compressor performance maps or ducting pressure losses. The effect of modeling error of the onboard model on the performance of the MPC will be explored later in this chapter.

Modeling error and simplification are cases when the model of the system does not match the actual performance of the system. Degradation, or faults, on the other hand, occurs when the performance of the system deviates from expected performance. In the case of degradation, as the system operates over time, such factors as erosion or fouling cause component performance, and in turn system performance, to change. A fault occurs when a component of the system is damaged and the performance changes. In both these cases the performance on the onboard model may not match the actual system

7.2.2.1 States on Tip-Jet Reaction Drive System

As discussed previously, the mechanisms which drive disconnects between the onboard model and the actual system are all different, however, the method used to bridge these disconnects is the same for all the mechanisms. The method used involves applying scalars to the airflow and efficiency values of the different components, such as compressors and turbines(CHATTERJEE and LITT 2003). These scalars can then be adjusted until the performance of the model matches the measured data of the system. Given that the onboard model matches the measured performance of the system, it is assumed that there should be a decent match of the unmeasured performance parameters.

The scalars used to match the measured parameters of the system act as additional states on the system. The data matching scalars will be defined by the vector, $\hat{p}$. In contrast to the states comprising the $\hat{x}$ vector of the linear state dynamics model, the data

match state is assumed to be steady state (i.e., the $\hat{p}$(i) value is the same as $\hat{p}(1)$). The state dynamics model using the measured output vector, y, can be updated to include the data matching scalars as shown in equations 122 and 123.

$$\hat{x}(t+1) = A\hat{x}(t) + B\hat{u}(t) + E\hat{p} \quad [126]$$

$$\hat{y}(t+1) = C\hat{x}(t+1) + F\hat{p} \quad [127]$$

These state dynamics equations can be simplified using an augmented state vector including both the original state vector and the data match scalars as shown in equation 124 and 125(CHATTERJEE and LITT 2003).

$$\hat{x}_{aug}(t+1) = A_{aug}\hat{x}_{aug}(t) + B_{aug}\hat{u}(t) \quad [128]$$

$$\hat{y}(t+1) = C_{aug}\hat{x}_{aug}(t+1) \quad [129]$$

Where state vector and matrices are augmented as follows:

$$\hat{x}_{aug} = \begin{bmatrix} \hat{x} \\ \hat{p} \end{bmatrix} \quad [130]$$

$$A_{aug} = \begin{bmatrix} A & E \\ 0 & I \end{bmatrix} \quad [131]$$

$$B_{aug} = \begin{bmatrix} B \\ 0 \end{bmatrix} \quad [132]$$

$$C_{aug} = \begin{bmatrix} C & F \end{bmatrix} \quad [133]$$

Due to the fact that the data matching scalars are not measured, a state estimator is needed to approximate them. Table 16 below summarizes all the potential states on the tip-jet reaction drive system that would need to be estimated using the state estimator. In addition to the data matching scalars, the three states of the $\hat{x}$ vector associated with the heat soak are also immeasurable

Table 16: Available States for Use in State Estimator

State, x	Data Match Scalar, p
Right LP Shaft Speed	Right Fan Flow
Right HP Shaft Speed	Right Fan Efficiency
Right Burner Metal Temperature	Right Compressor Flow
Left LP Shaft Speed	Right Compressor Efficiency
Left HP Shaft Speed	Right HP turbine Flow
Left Burner Metal Temperature	Right HP Turbine Efficiency
Tip-jet Rotor Shaft Speed	Right LP turbine Flow
Tip-jet Burner Metal Temperature	Right LP Turbine Efficiency
	Left Fan Flow
	Left Fan Efficiency
	Left Compressor Flow
	Left Compressor Efficiency
	Left HP turbine Flow
	Left HP Turbine Efficiency
	Left LP turbine Flow
	Left LP Turbine Efficiency

7.2.2.2 Sensors on a Tip-Jet Reaction Drive System

To accurately estimate the augmented state vector, there needs to be as many measured values as there are states. Figure 40 below shows the potential location of the sensors on the tip-jet reaction drive system. Included in the sensor suite are typical gas turbine gas path pressure, temperature, and shaft speed measurements (ADIBHATLA and GASTINEAU 1994) as well as hub pressure and tip-jet shaft speed. Although not shown in this chart, there is a HP and LP turbine that are connected via a shaft to the compressor and fan, respectively. Both the HP and LP shaft operate at different rotational speeds and have a separate measurement.

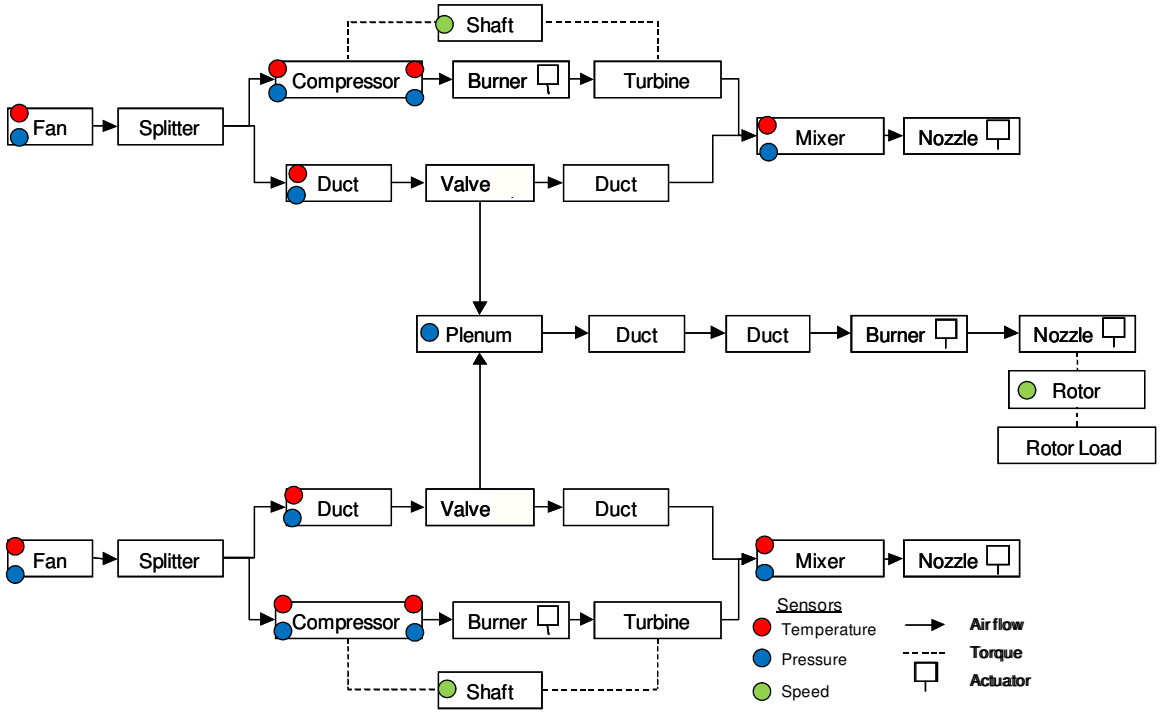

Figure 40: Tip-Jet Reaction Drive System Schematic with Sensor Locations

Table 17 summarizes the available sensors for use in the state estimator for the tip-jet reaction drive system. In the state estimator, fan inlet pressure and temperature are considered boundary conditions and are not included in the list. From this list and the available states in Table 16, the state estimator that will be used on the MPC for the tip-jet reaction drive system can be designed. Of note there are only 22 sensors available to estimate 24 states. Therefore, even if all the all available sensors could be used in the state estimator, not all the states can be observed.

Table 17: Available Sensors for Tip-Jet Reaction Drive System

Measured Output, y
Right Bypass Duct Temperature
Right Bypass Duct Pressure
Right Compressor Inlet Temperature
Right Compressor Inlet Pressure
Right Compressor Exit Temperature
Right Compressor Exit Pressure
Right LP Turbine Exit Temperature
Right LP Turbine Exit Pressure
Right LP Shaft Speed
Right HP Shaft Speed
Left Bypass Duct Temperature
Left Bypass Duct Pressure
Left Compressor Inlet Temperature
Left Compressor Inlet Pressure
Left Compressor Exit Temperature
Left Compressor Exit Pressure
Left LP Turbine Exit Temperature
Left LP Turbine Exit Pressure
Left LP Shaft Speed
Left HP Shaft Speed
Hub Pressure
Tip-jet Rotor Shaft Speed

7.2.2.3 State Estimator State and Sensor Selection

An invaluable method used for selecting the appropriate state and sensors used in the state estimator is singular value decomposition (SVD) (STRANG 2006). The SVD decomposes a matrix into three components as shown below.

$$A = U\Sigma V^T \quad [134]$$

The matrices U and V map the outputs and inputs of A, respectively, onto an orthonormal basis. The matrix Σ is diagonal and contains the square roots of the nonzero eigenvalues of A^TA and AA^T. Each of the positive entries along the diagonal of Σ is known as a singular value. Similar to the RGA methodology used in the control

input/output selection, the SVD can be used to capture the magnitude of the relationship between inputs and outputs.

Assuming a fixed control input, the augmented state dynamics model can be simplified to show the transient augmented state/sensor relationship as shown below

$$\hat{y} = C_{aug} A_{aug} \hat{x}_{aug} \quad [135]$$

By substituting equation 134 into 135, the SVD of the transient augmented state/sensor relationship of the system is given by (dropping the aug subscript for simplicity):

$$\hat{y} = U\Sigma V^T \hat{x} \quad [136]$$

Before any further analysis on the state/sensor relationship matrix is done, a quick check on the invertibility of the matrix is required. If the matrix is not very invertible, states or sensors may have to be removed. A good way to perform this check is to calculate the condition number of the matrix. The condition number is defined as the ratio of the maximum singular value of Σ to the smallest singular value of Σ. The larger the condition number the less invertible the matrix is. The large condition number is often seen when the minimum singular value approaches zero which leads to a divide by zero condition.

The condition number of the augmented state/sensor matrix using the available states and sensors from Table 16 and Table 17 is infinity. By eliminating duct temperature, duct pressures, and hub pressure, the condition number of the augmented state/sensor matrix can be reduced to approximately 14. However, this reduces the number of available sensors to 17, which means only 17 of the 24 states can be estimated. From here further analysis can be performed to define which of the 17 states can be best estimated.

Expanding the U and V into columns vectors, and Σ into singular values, equation 136 can be rewritten as follows.

$$\hat{y} = \begin{bmatrix} U_1 & \cdots & U_m \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & \sigma_m & 0 \end{bmatrix} \begin{bmatrix} V_1 & \cdots & V_n \end{bmatrix}^T \hat{x} \qquad [137]$$

Using column vectors of V and the singular values of Σ, an observability index for each state can be calculated as follows (BRUNELL, VIASSOLO and PRASANTH 2004):

$$OI_i = V_i \sqrt{\Sigma^T \Sigma} V_i^T \qquad [138]$$

Figure 41 below shows observability indices for all the augmented states using the 17 available sensors. Seven states are significantly less observable than any others: tip-jet burner heat soak, right engine HP efficiency, right engine LP efficiency, right engine LP airflow, left engine HP efficiency, left engine LP efficiency, and left engine LP airflow. Tip-jet burner heat soak is not observable because its dynamics are too fast and besides tip-jet shaft speed there are no other sensors physically located near the state. All the data matching scalars on the turbine besides HP airflow are not observable because there is not any turbine inlet or inter-stage pressure or temperature sensors. These sensors are not available because of the extremely high temperatures of the gas path.

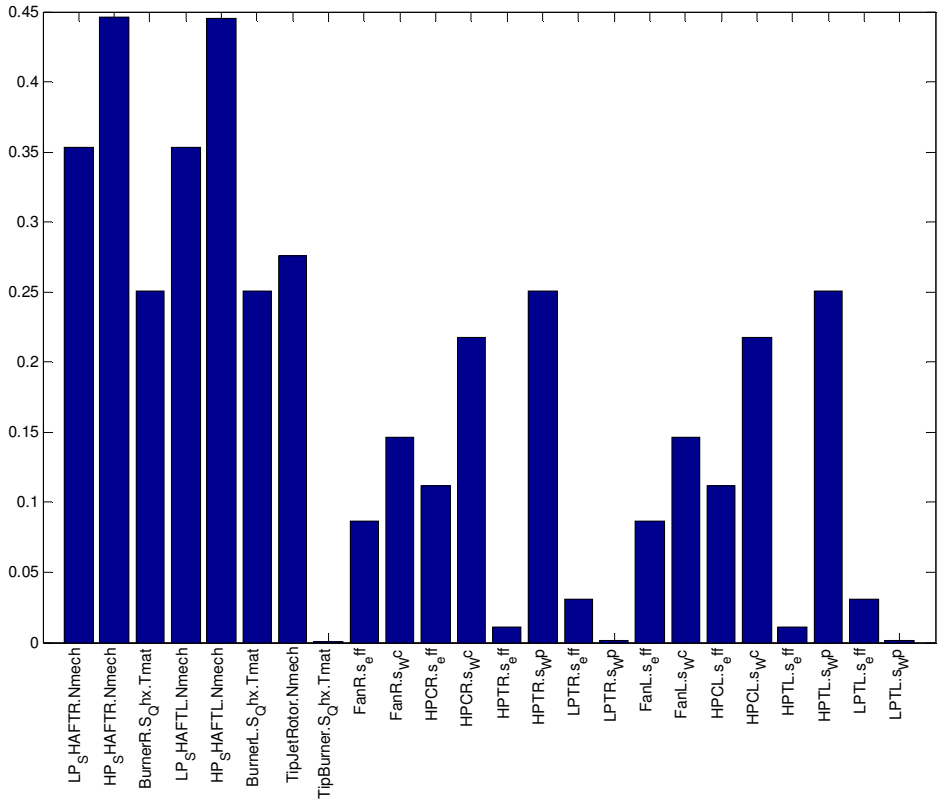

Figure 41: State Observability Indices

It is also important that the states being observed have a significant effect on the control parameters of the MPC. Or conversely, it is important to understand if an unobservable state has a significant effect on the control parameters. The performance of the MPC may be adversely affected if unobservable states have a strong relationship with the control parameters.

. Similar to the above observability study, a sensitivity study using the control parameters of the $\hat{z}$ vector. Again assuming a fixed control input, the augmented state dynamics model can be simplified to show the transient augmented state/control parameter relationship as follows.

$$\hat{z} = C_{z,aug} A_{aug} \hat{x}_{aug} \qquad [139]$$

Using SVD the simplified transient model can be decomposed as follows.

$$\hat{z} = U_z \Sigma_z V_z^T \hat{x} \qquad [140]$$

Analogous to the observability indices, sensitivity indices for each state can be calculated.

$$SI_i = V_{z,i} \sqrt{\Sigma_z^T \Sigma_z} V_{z,i}^T \qquad [141]$$

The sensitivity indices are shown in Figure 42. The unobservable tip-jet heat soak does not affect the output, but all the other system states have a large effect. The data matching scalars that are most sensitive are all very observable.

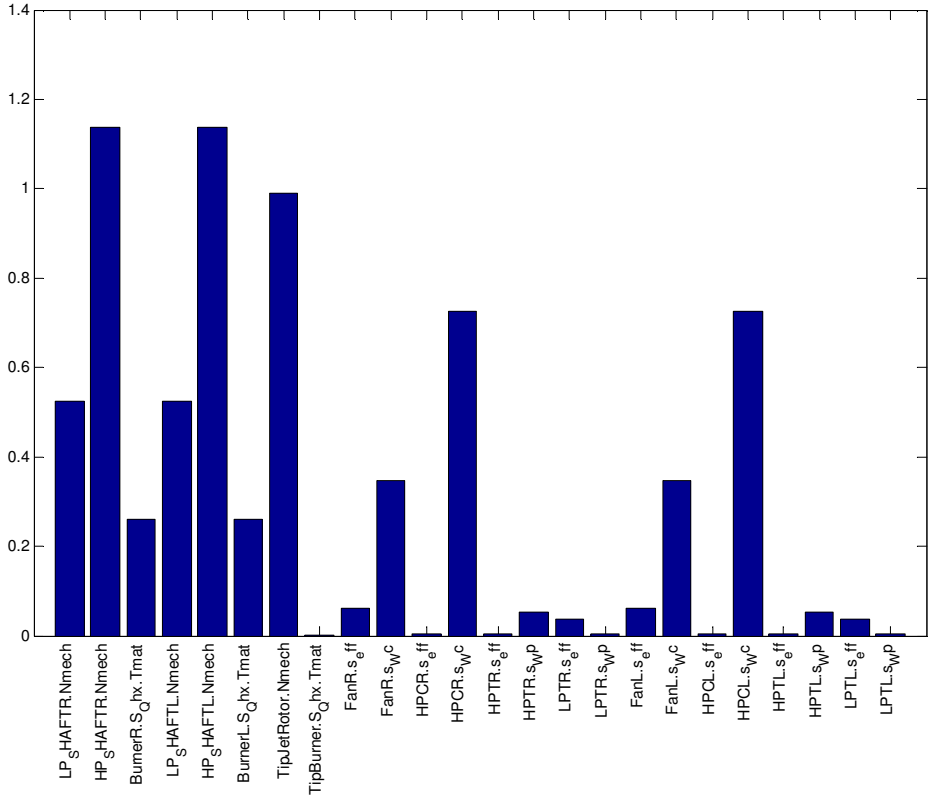

Figure 42: State Sensitivity Indices

From the above analysis the 17 sensors used in the state estimator were selected. Additionally the most observable states were identified. There are seven system states and ten data match scalars. Table 18 summarizes the final choice of the states and sensors used for the state estimator of the tip-jet reaction drive MPC architecture.

Table 18: States and Sensors Used in Tip-Jet Reaction Drive State Estimator

State, x	Data Match Scalar, p	Measured Output, y
Right LP Shaft Speed	Right Fan Flow	Right Compressor Inlet Temperature
Right HP Shaft Speed	Right Fan Efficiency	Right Compressor Inlet Pressure
Right Burner Metal Temperature	Right Compressor Flow	Right Compressor Exit Temperature
Left LP Shaft Speed	Right Compressor Efficiency	Right Compressor Exit Pressure
Left HP Shaft Speed	Right HP turbine Flow	Right LP Turbine Exit Temperature
Left Burner Metal Temperature	Left Fan Flow	Right LP Turbine Exit Pressure
Tip-jet Rotor Shaft Speed	Left Fan Efficiency	Right LP Shaft Speed
	Left Compressor Flow	Right HP Shaft Speed
	Left Compressor Efficiency	Left Compressor Inlet Temperature
	Left HP turbine Flow	Left Compressor Inlet Pressure
		Left Compressor Exit Temperature
		Left Compressor Exit Pressure
		Left LP Turbine Exit Temperature
		Left LP Turbine Exit Pressure
		Left LP Shaft Speed
		Left HP Shaft Speed
		Tip-jet Rotor Shaft Speed

A sensitivity study can be performed on the selected sensors and states to ensure that all the sensors are fairly sensitive to changes in the selected state (BORGUET and LEONARD 2008). Figure 43 below shows the sensitivity study for the selected sensors and states. This type of analysis could be used to study potential new sensors or compare different sensor combinations.

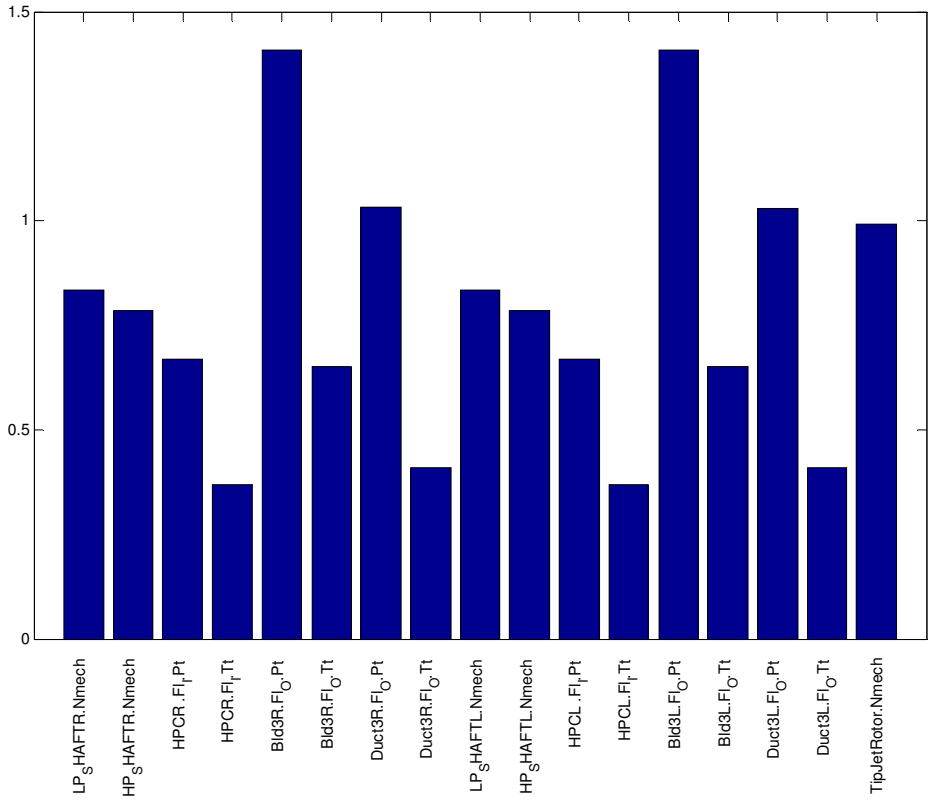

Figure 43: Sensor Observability Indices

7.2.2.4 Extended Kalman Filter

The Extended Kalman Filter (EKF) is a popular method for performing data matching and state estimation on nonlinear systems like the tip-jet reaction drive system. The extended in EKF implies that the nonlinear system is linearized around the current estimated operating point. In a linear Kalman Filter, the error covariance of the estimate of the state, P_k, is minimized by applying an optimal gain, known as the Kalman gain, to the difference between the linear estimate of the measured output and the actual measurements in the presence of both state and measurement uncertainty(GUSTAFSSON 2000).

State and measurement uncertainty are both included in the state dynamics equations as follows:

$$\hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k + Q_k \quad [142]$$

$$\hat{y}_k = C\hat{x}_k + R_k \quad [143]$$

Where Q_k is the covariance of the state uncertainty and R_k is the covariance of the sensors. The Kalman Filter uses a predictor-corrector method to estimate the state, where the state and measurements are first estimated using the linear model.

$$\hat{x}_{k+1}^- = A\hat{x}_k + B\hat{u}_k \quad [144]$$

$$\hat{y}_k = C\hat{x}_k \quad [145]$$

A predictor-corrector method is also used for the estimation of the error covariance.

$$P_{k+1}^- = AP_kA^T + Q_k \quad [146]$$

A residual comparing the estimate of the measured output with the actual measured output is defined as follows:

$$\hat{e}_k = \bar{y}_k - \hat{y}_k \quad [147]$$

Where the covariance of the measurement residual can then be defined

$$S_k = CP_{k+1}^-C^T + R_k \quad [148]$$

The Kalman gain is defined such that the estimate of P_{k+1} is minimized

$$K_k = P_{k+1}^-C^TS^{-1} \quad [149]$$

Using the Kalman gain and the measurement residual, the corrected state estimate is calculated

$$\hat{x}_{k+1} = \hat{x}_{k+1}^- + K_k\hat{e}_k \quad [150]$$

In a similar fashion, the error covariance is corrected as shown below.

$$P_{k+1} = P_{k+1}^- - K_kCP_{k+1}^- \quad [151]$$

Combining equations144, 145, 147, and 150 together, the calculation of the estimate of the state can be shown as a function of the three inputs to the state estimator.

$$\hat{x}_{k+1} = (A - K_k CA)\hat{x}_k + (I - K_k C)B\hat{u}_k + K_k \bar{y}_{k+1} \quad [152]$$

7.2.3 Model Predictive Control Algorithm

Now that the inputs and outputs to the MPC have been defined, and the state estimator has been designed, various variables in the MPC algorithm can be defined. The MPC control algorithm defined below is inherently different from a PI controller (both model and non-model based). The PI controller transposes the difference between a reference value and a measured or estimated parameter at the current time-step, in order to set the control inputs. Whereas the MPC controller determines the control inputs by minimizing a cost function which the sum of future difference between the reference value and the measured or estimated parameter over a prediction horizon. This cost function may also include a term to minimize the energy of the control over a control horizon. The effect of this term on the response of a MPC is similar to an integral gain on a PI controller. The algorithm for a centralized MPC is defined as follows:

$$V(x) = \sum_{i=Hw}^{H_p} \left\| \hat{z}(k+i/k) - r(k+i) \right\|_{Q(i)}^2 + \sum_{i=0}^{H_u - 1} \left\| \Delta\hat{u}(k+i/k) \right\|_{R(i)}^2$$

Where

- $\hat{z}$ are model calculated parameters
- $\hat{r}$ are reference or target values
- $\hat{u}$ are actuators positions
- $\hat{v}$ are measured disturbances
- H_p is thde prediction horizon
- H_u is the control horizon
- H_w is the prediction horizon start point
- Q and R are weighting matrices

Subject to these constraints

$$const < \hat{z}(t) < const$$
$$0 < \Delta\hat{u}(t) < const$$
$$const < \hat{u}(t) < const$$

Governed by state dynamics model

$$\hat{x}(t+1) = A\hat{x}(t) + B\hat{u}(t) + B_d\hat{v}(t)$$
$$\hat{z}(t+1) = C\hat{x}(t+1) + D_d\hat{v}(t)$$

Governed by control input equations

$$\hat{u}(t) = \hat{u}(t-1) + \Delta\hat{u}(t)$$

The onboard model can be directly incorporated into the MPC algorithm by inserting the linear state dynamics model and the control input equations into the cost function. For non-model based PI controllers, the model is used offline to develop the control gains. MPC also directly integrates constraints on system outputs, control input position, and control input rate change into the cost function by using Lagrange multipliers.

Given the above definition of the MPC algorithm, the understanding of tip-jet reaction drive system dynamics, selection of the model based control inputs and outputs, and the knowledge of which system states can be observed, some components of the MPC algorithm can be quantified. The steady state model based design can be directly used to define the variables in the r and z vectors of the cost function. In addition, the understanding of the system dynamics can also be used to define the prediction horizon, control horizon, and the model time-step. Furthermore, the state estimator design has defined which states can be observed, thus allowing them to be variables in the cost function of MPC.

Other MPC algorithm components such as the weighting matrices have significant effect on control performance and can be viewed as design variables. Although not considered a design variable, constraint handling is an important part of the MPC algorithm. Both design variables and constraint handling are also introduced in the following section and will be explored in more detail during the control sizing and simulation sections later in the chapter.

7.2.3.1 Control Input, Output, Reference, State Definition

The analysis in Section 7.1.1 was used to define the variables that populate the $\hat{r}$ and $\hat{z}$ vectors of the MPC model. These variables were: left engine thrust, left engine CEPR, right engine thrust, right engine CEPR, tip-jet shaft speed, and hub pressure. The control inputs populating the $\hat{u}$ vector were: left engine fuel flow, left engine exhaust area, right engine fuel flow, right engine exhaust area, tip-jet fuel flow, and tip-jet exhaust nozzle area. From the analysis of the state estimator, all shaft speeds and engine heat soaks will be used as the states in the state dynamics model. Additionally, the rotor load and all the estimated data match scalars are considered measured disturbances in the state dynamics model (MUSKE and BADGWELL 2002). Table 19 below summarizes all the control output parameters, control inputs, system states, and measured disturbances that will be used in the MPC of the tip-jet reaction drive system.

Table 19: Model Based Control Inputs, Outputs, References, States, and Disturbances

Control Output and Reference, z and r	Control Inputs, u	State, x	Measured Disturbances
Right Engine Thrust	Right Engine Fuel Flow	Right LP Shaft Speed	Rotor Load
Right Engine CEPR	Right Engine Exhaust Nozzle	Right HP Shaft Speed	Right Fan Flow
Left Engine Thrust	Left Engine Fuel Flow	Right Burner Metal Temperature	Right Fan Efficiency
Left Engine CEPR	Left Engine Exhaust Nozzle	Left LP Shaft Speed	Right Compressor Flow
Tip-Jet Rotor Shaft Speed	Tip-jet Fuel Flow	Left HP Shaft Speed	Right Compressor Efficiency
Hub Pressure	Tip-jet Exhaust Nozzle	Left Burner Metal Temperature	Right HP turbine Flow
		Tip-jet Rotor Shaft Speed	Left Fan Flow
			Left Fan Efficiency
			Left Compressor Flow
			Left Compressor Efficiency
			Left HP turbine Flow

Over the prediction horizon, it is not always important that the estimated output matches the reference value. For example, when the engine throttle position is changed, it is not expected that the engine speed or thrust match the demand until approximately a similar time on the order of the time constant of the rotor. To account for this, the MPC algorithm includes a design variable, H_w, that moves the reference matching point in the prediction horizon to a later time than the first prediction time-step. If this variable were

not included, the order of the model/reference error in the first few prediction time-steps may dominate the cost function resulting in large control moves and a potentially unstable control. The effect of this variable on the control performance will be explored in the MPC control sizing section of this chapter.

7.2.3.2 Prediction Horizon and Prediction Time-step

The analysis of the dynamics done in section 6.2.1 can be used to define the number of prediction and control horizon steps as well as the sampling time-step of the simulation. The sampling time-step should not be much larger than the fastest dynamic of the system; otherwise the model used for MPC may not be accurate enough. Using the dynamics defined in

Table 7, the fastest dynamic time constant is approximately 0.1s. Using the factor of 5 or 10 to ensure accurate modeling fidelity, the model time-step should be around 0.02s.

To accurately capture system response, the prediction horizon must be longer than the time constant of the slowest system dynamic. If the prediction horizon is too short, the resulting control may become unstable or provide suboptimal performance. From the dynamic analysis in Table 7, the slowest dynamic time constant is around 2s, thus requiring the prediction horizon to be at least 2s. Combining the prediction horizon time requirement and the model time-step, the number of prediction horizon steps should be on the order of 100.

The number of the control horizon steps should be of similar order to the prediction horizon. However, a blocking scheme or a shorter control horizon would result in a smaller number of control horizon steps. For example in the proposed MPC algorithm discussed in the next chapter, each control input is optimized every other time-

step, necessitating an every other time-step blocking scheme. The effect of the control horizon length on the performance of the centralized MPC will be explored in section 7.3.

The tip-jet reaction drive system provides a very challenging MPC computational problem because the dynamics of interest have a wide range. The model time-step requirement is defined by the engine rotor and heat-soak dynamics, whereas the prediction horizon time is defined by the tip-jet rotor dynamics. If the MPC problem was limited to just the engine, the prediction horizon would be approximately 0.1 to 0.2s, requiring only about 20 prediction horizon steps and have a significantly reduced computational burden.

7.2.3.3 Q and R Weighting Factors

The Q and R weighting factors in the cost function are variables which emphasize the importance of matching the reference target versus minimizing control input movements. For the MPC algorithm to be stable and solvable the values of the weighting factors need to be greater than or equal to zero for all points along the prediction and control horizon.

As mentioned previously, Q is a weighting factor applied to the reference matching portion of the MPC cost function. Besides the greater than or equal to zero constraint, Q can hold any value. For this set of experiments Q was assumed to be time variant. However to limit the number of potential variables the same weighting factor is to be applied to each control parameter, j. Q can also be used to capture to effect of H_w by setting all values of Q(i) equal to zero for all i that are less than H_w.

R is a weighting factor on the movements of the control input portion of the MPC cost function. Similar to Q, the only fixed criteria for R is that is be greater than or equal to zero for points along the control horizon. R acts as a damper on the response where the larger the value of R, the slower the response.

In addition to the absolute values of R and Q, it is equally important to consider the relative value of each when compared with each other. As the value of R becomes significantly larger than Q, the control moves much slower to the extent that the control inputs would not move at all if R was too much larger than Q. In the MPC sizing section in section 7.3, various combinations of Q and R, as well as values of H_w will be explored to obtain the desired MPC performance.

7.2.3.4 Tip-jet Reaction Drive Constraints

The constraints on the tip-jet jet reaction drive system parallel the structure of the MPC cost function. The first set of constraints is related to the performance of the system, while the second set of constraints is related to the movement and position of the control inputs.

These first set of constraints range from but are not limited to speed, temperature, and pressure limits. Exceeding limits such as minimum or maximum shaft speed or maximum turbine inlet temperature may not cause an immediate change in system performance but would lead to a reduction in the useful life of the system. Whereas exceeding a stall margin limit could potentially cause the system to fail catastrophically. For a PI type control, a separate controller is required for each constraint, however for MPC the constraints are built directly in the algorithm. Therefore, only a single control is needed to handle all the performance constraints.

The second set of constraints is defined by the physical limitations of the actuators used on the system. First there is a maximum and minimum position of the actuator. And secondly there is a maximum rate of change of the position of the actuator. Whenever these limits are reached, the situation is defined as integrator windup. To overcome this issue in control architectures other than MPC, an additional windup protection scheme is required (KRISHNAKUMAR, NARAYANSWAMY and GARG 1996). In contrast, these constraints are directly handled in the MPC algorithm.

As the system operates over the flight envelope, the system may operate at or exceed the boundary of these constraints. Therefore the MPC algorithm needs to account for potentially all of these constraints. To understand how the MPC handles constraints, simulations will be run where some constraints become active, with particular focus on the fan stall margin and actuator rate limit constraints.

7.2.4 Centralized MPC Modeling and Simulation Environment

Using the architecture shown in Figure 39 as a guideline, the centralized MPC modeling and simulation (M&S) environment was developed using Simulink. For this controller architecture, there are four major components that need to be modeled: the tip-jet reaction drive system, G(s), which transposes control inputs to measured output, the onboard model, $G_{ob}(s)$, which is used to estimate unmeasurable system parameters, the state estimator which is used to both estimate the states of the system and ensure the onboard model matches the measured parameters of the system, and centralized MPC which determines the control inputs required to minimize a defined objective function. As was done for the PI controller, the tip-jet reaction drive system is represented by the non-linear NPSS defined earlier in section 6.2.1.3. The NPSS model is executed from within Simulink using a custom dynamic link library (DLL) developed by NASA. All the inputs and outputs into this model are normalized around the design point to ensure good matrix properties for the controller design.

The onboard model is represented by the linearized version of the normalized inputs and outputs of the NPSS model. Normally, multiple linear models would be created representing multiple operating points and power settings with these models the being curve fit together. However since the analysis is done around a single operating point, only a single linear model was used for the onboard model.

The state estimator uses the extended Kalman filter defined in section 7.2.2.4 to estimate the state of the tip-jet reaction drive system. As was discussed in the state

estimator design section performed in section 7.2.2.3, the state estimator uses the linearized onboard model, but this model will be augmented by data match scalar states. This ensures that both the onboard model output matches the actual output of the tip-jet reaction drive system and the controller has offset free reference tracking. To simulate the state estimator, a custom Simulink S-Function, whose source code is located in Appendix D, was created that contained all the extended Kalman filter logic.

The centralized MPC controller also uses the linearized onboard model augmented by data match scalars. The centralized MPC controller was simulated using a Simulink MPC element. The execution of the Simulink based centralized MPC M&S environment is done from a Matlab command prompt. Now that the all the components of the centralized MPC architecture have been defined and the M&S environment has been developed, the centralized MPC can be sized.

7.3 Centralized Model Predictive Control Design

The metrics for the centralized MPC controller performance are virtually identical to that of the PI controller. For engine thrust and CEPR the target bandwidth is approximately 3 rad/s. For the tip-jet rotor speed, the target bandwidth is around 0.6 rad/s. The hub pressure bandwidth should be similar to the engine bandwidth of 3 rad/s. The interactions should be minimal across all frequencies. For stability concerns, gain margin and phase margin should be greater than 6 dB and 45^{o}, respectively.

7.3.1 Control Sensitivity to Design Variables

As mentioned in the previous section, the centralized MPC design variables are the prediction horizon start point for each variable in the cost function, the cost function weighting factors Q and R, and the control horizon. The sensitivity of the control performance metrics to each of these variables must be analyzed. Before the sensitivity study is performed, a high level analysis is performed to find a satisfactory starting point

for each design variables. From this point, each of the design variables can be changed and the results analyzed with the goal of finding a design that meets the control performance metrics. For reference, the baseline point values for each design variables are: $H_{we} = 15$, $H_{wr} == 80$, $R = 2.5$, $Q = 0.5$, $H_p = 100$, and $H_u = 100$.

As was noted in the PI controller design section, each column in the Bode plot represents the change in demand, whereas each row represents the response of the parameter to a change in demand. Similar to the PI controller, shorthand notation as shown in Table 20 will be used to capture the response of different control parameters to changes in demand. And was done in the previous chapter, the green circles will highlight the bandwidth sensitivity while the red squares highlight the largest interactions.

Table 20: Centralized MPC Bode Plot Descriptions

Bode Plot Column and Row Description	Column Description	Row Description
Fg Right	Right engine thrust demand change	Right engine thrust response to a demand change
CEPR Right	Right engine CEPR demand change	Right engine CEPR response to a demand change
Fg Left	Left engine thrust demand change	Left engine thrust response to a demand change
CEPR Left	Left engine CEPR demand change	Left engine CEPR response to a demand change
Tip-jet Shaft	Tip-jet shaft speed demand change	Tip-jet shaft speed response to a demand change
Hub Pressure	Hub pressure demand change	Hub pressure response to a demand change

7.3.1.1 Prediction Horizon Start Point

The first MPC design variable to be explored is the prediction horizon start point. The prediction horizon start point is the value of the first time-step included in the MPC cost function. Each of the control variables such as thrust and CEPR can have a different prediction horizon start point. However, to keep the analysis simpler, prediction horizon

start point was grouped into two categories: engine and rotor. The engine prediction horizon start point contains engine thrusts, engine CEPRs, and hub pressure while the rotor prediction horizon start point contains just the tip-jet rotor shaft speed. The variables were grouped as such because of the target bandwidth of the controller was divided similarly.

Figure 44 and Figure 45 below show Bode plot sensitivity to changes in the engine prediction horizon start point, H_{we}. The engine prediction horizon start point ranged from a value of one to 80. As a point of reference, the time constant for the engine shaft speeds is approximately 20 time-steps. For all the control except tip-jet rotor speed, as H_{we} increases, the bandwidth decreases. The engine bandwidth ranged from 0.7 to 3.4 rad/s. The CEPR bandwidth ranged from 0.5 to 6.2 rad/s. While the hub pressure ranged from 0.3 to 1.8 rad/s. At an engine prediction start point value of 15, the engine, CEPR, and hub pressure bandwidths are 2.4, 2.8, and 1.4, respectively. Using just prediction horizon start point, the hub pressure cannot achieve the target bandwidth. However these values are fairly close to the target values. The value of 15 for H_{we} is about ¾ the size of the engine time constant.

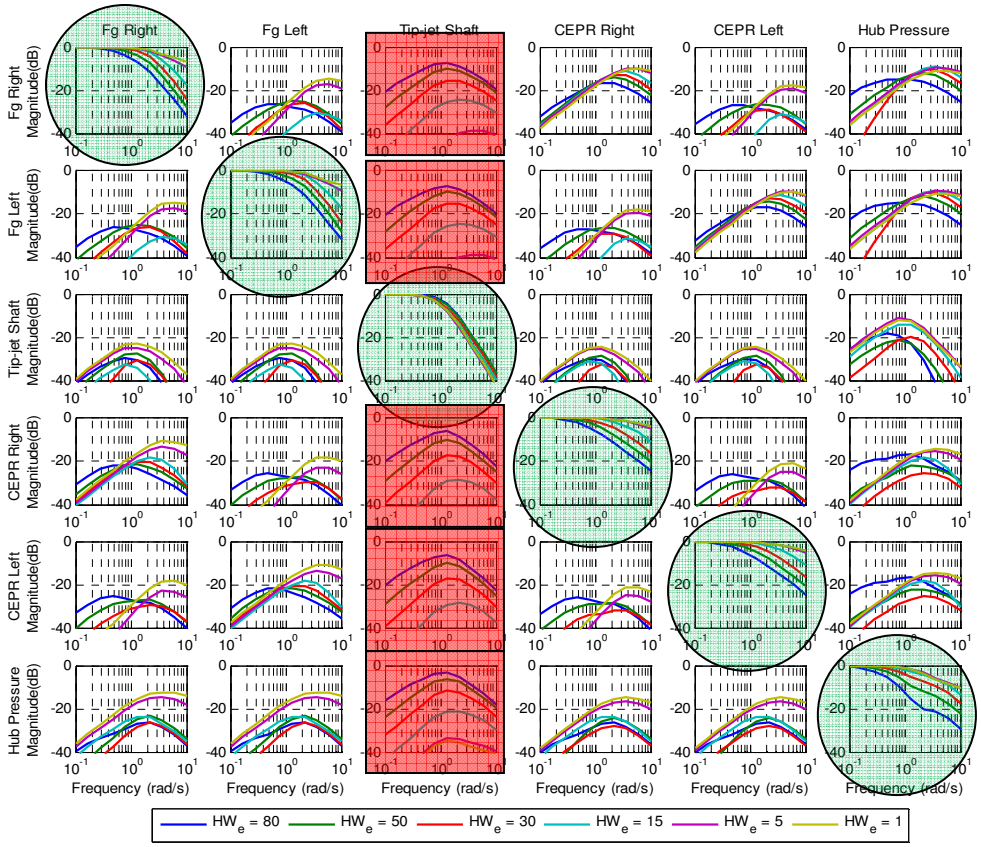

Figure 44: Bode Plot Magnitude Sensitivity to Changes in Engine Prediction Horizon Start Point

As shown in Figure 44, for all design points tested, the interactions did not become significant. However at high values of H_{we}, all the other control parameters had more noticeable interactions with changes in tip-jet shaft speed demand. H_{we} did not have much of an effect on the phase margin. As H_{we} decreased, CEPR and hub pressure gain margin decreased from about 40 to 20 dB. On the other hand, engine gain margin had the opposite effect by increasing from 20 to 30 dB as H_{we} increased. In all these cases, the gain and phase margin exceeded the target values.

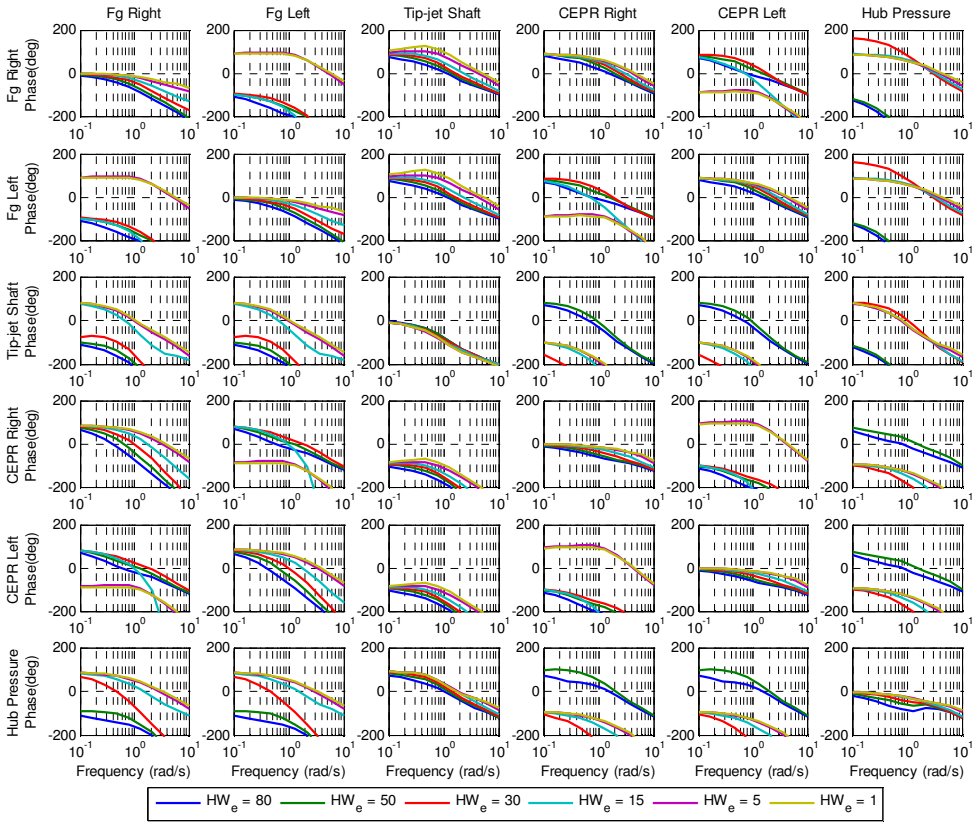

Figure 45: Bode Plot Phase Sensitivity to Changes in Engine Prediction Horizon Start Point

Figure 46 and Figure 47 below show Bode plot sensitivity to changes in the rotor prediction horizon start point, H_{wr}. Using the engine prediction horizon start point analysis as a reference point, the range for H_{wr} was chosen to be around ¾ of the tip-jet rotor speed time constant of 100 time-steps. As expected the only control parameter bandwidth to be significantly affected by H_{wr}. Similar to H_{we} , as H_{wr} increases, the bandwidth decreases. The rotor bandwidth ranged from 0.7 to 1.9 rad/s. At a rotor prediction start point value of 90, the rotor bandwidth 0.7 rad/s. Similar to hub pressure, using just prediction horizon start point, the rotor speed cannot achieve the target bandwidth but can get fairly close.

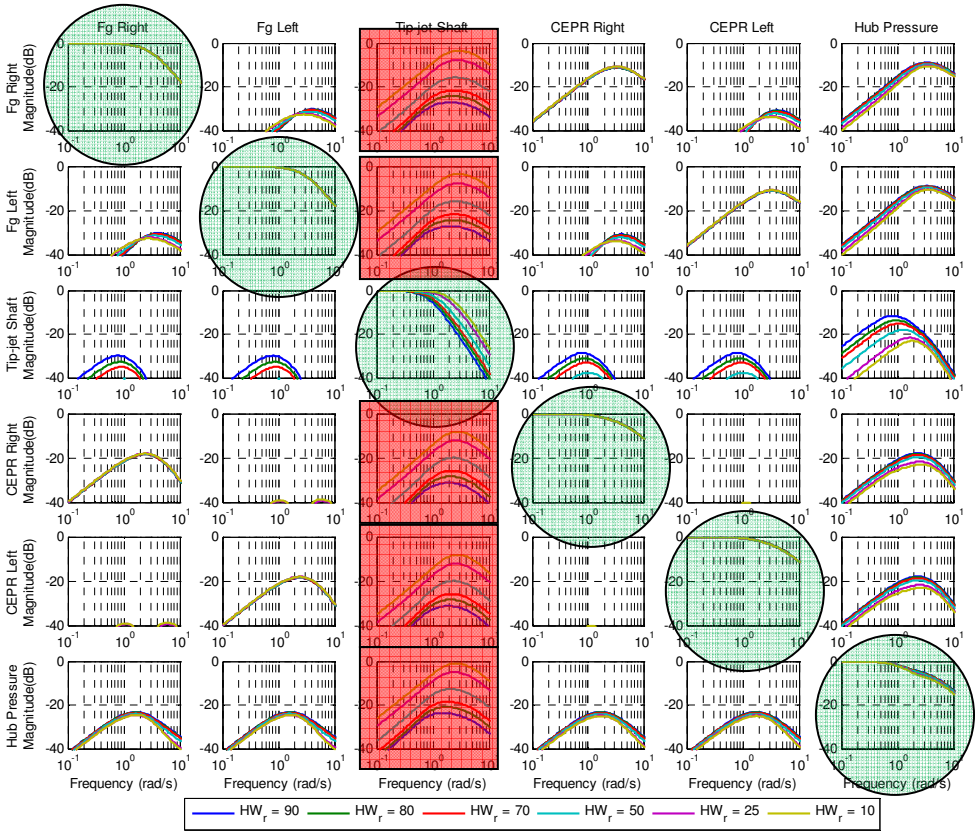

Figure 46: Bode Plot Magnitude Sensitivity to Changes in Rotor Prediction Horizon Start Point

As shown in Figure 46, at low values of H_{wr}, the interactions of all the other control parameters with tip-jet shaft speed demand become close to significant. However since low values of H_{wr} do not get close to the bandwidth targets, this should not be an issue. H_{wr} did not have much of an effect on the phase margin. As H_{we} decreased, tip-jet rotor speed gain margin decreased from about 27 to 20 dB. In all these cases, the gain and phase margin exceeded the target values.

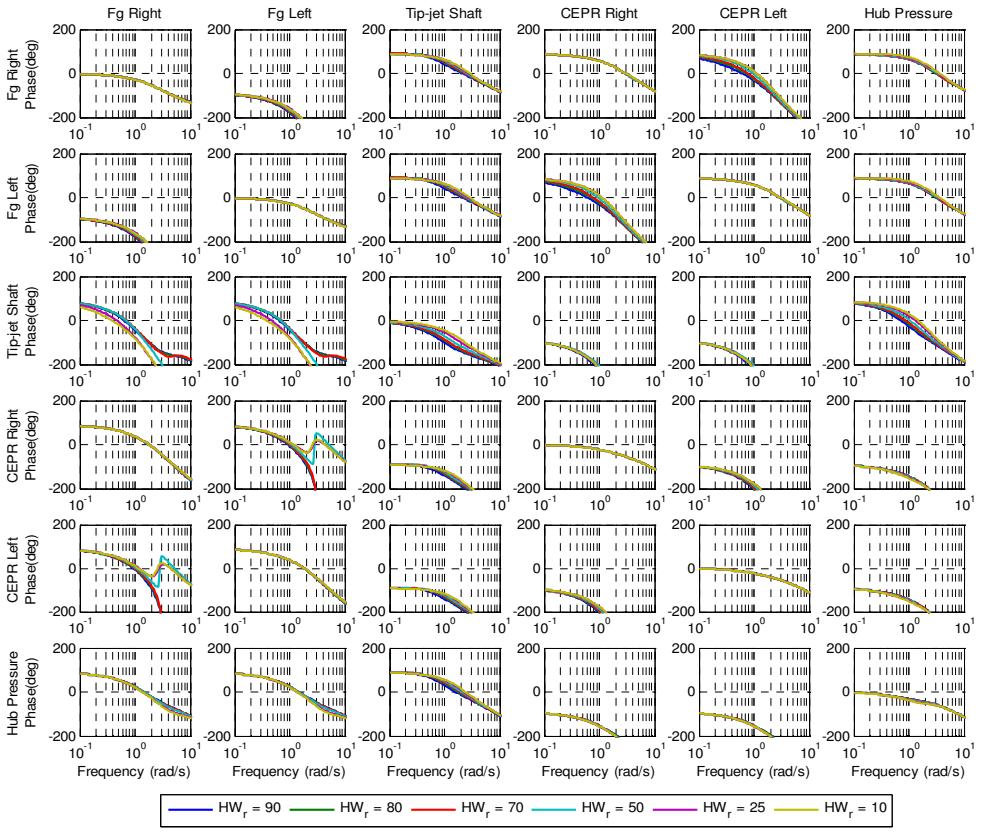

Figure 47: Bode Plot Phase Sensitivity to Changes in Rotor Prediction Horizon Start Point

7.3.1.2 Cost Function Weighting Factors

Figure 48 and Figure 49 below show Bode plot sensitivity to changes in the MPC cost function control movement weighting factor, R. The range of R was from 20 times as large a Q (R=20) to 5 times as small (R=0.1). As R increases, the bandwidth of the all parameters decreases. Engine thrust bandwidth ranged from 5 to 1 rad/s. Tip-jet rotor speed bandwidth varied from 1.2 to 0.3 rad/s. CEPR bandwidth ranged from 5 to 1.2 rad/s. While hub pressure bandwidth ranged from 11 to 0.3 rad/s. The target bandwidth value of all the control parameters falls within the explored range.

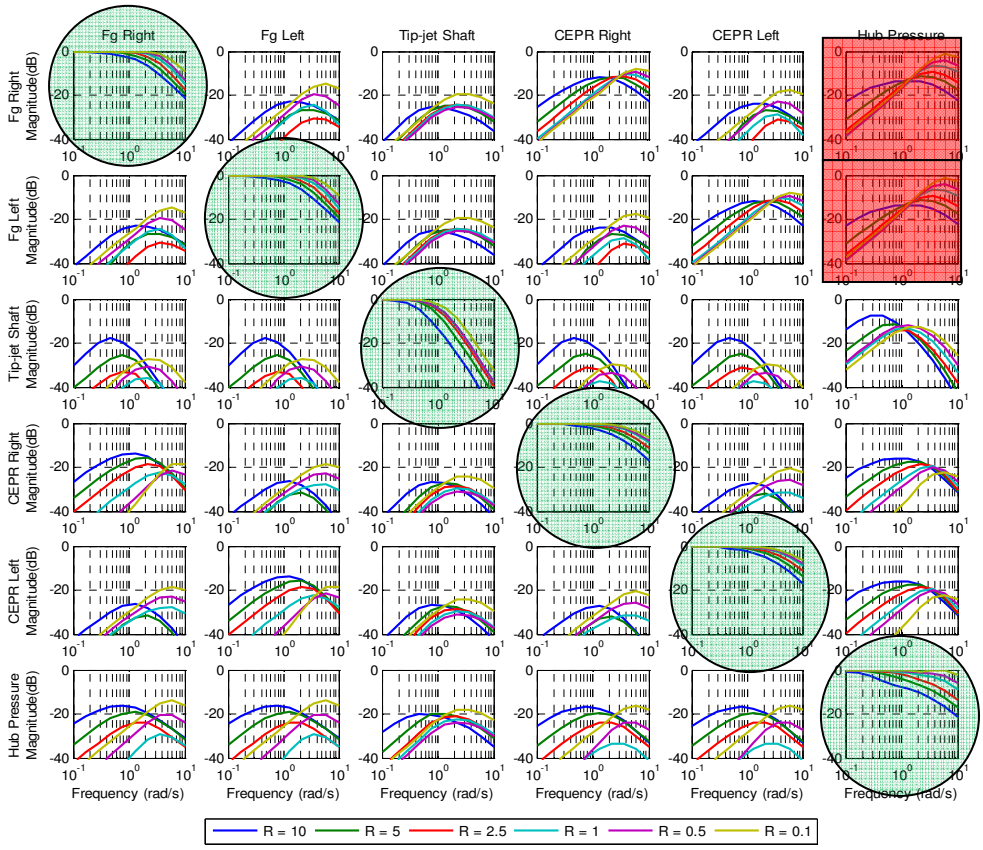

Figure 48: Bode Plot Magnitude Sensitivity to Changes in Control Movement Weighting Factor

As shown in Figure 48, for all design points tested, the interactions did not become significant. However for low values of R the high frequency interactions of engine thrust with changes in hub pressure demand become fairly noticeable. Even though the gain and phase margin exceeded the target values for all cases, R had a significant effect on the phase and gain margin. For all the control parameters, as the value of R decreases, both the phase and gain margin decreased. The engine thrust phase and gain margin saw the biggest changes and at very small values of R approached the limits. Engine thrust ranged from 135^{o} to 80^{o} and 50 to 9 dB, respectively. Tip-jet rotor speed ranged from 110^{o} to 100^{o} and 34 to 23 dB, respectively. Engine CEPR ranged from 138^{o} to 120^{o} and 28 to 20 dB, respectively. And hub pressure ranged from 150^{o} to 90^{o} and 33 to 16 dB, respectively.

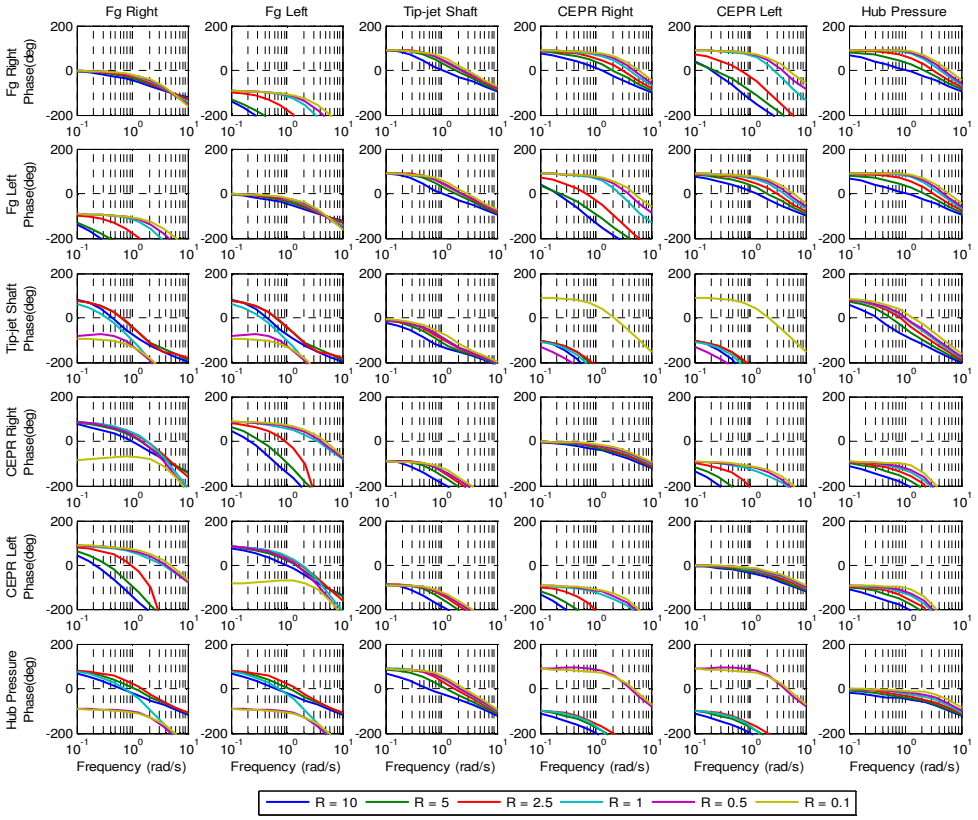

Figure 49: Bode Plot Phase Sensitivity to Changes in Control Movement Weighting Factor

Figure 50 and Figure 51 below show Bode plot sensitivity to changes in the MPC cost function reference tracking weighting factor, Q. The range of Q was from 2 times as large as R (Q=5) to 25 times as small (Q=0.1). As expected Q has the opposite effect as R. As Q increases, the bandwidth of the all parameters increases. Engine thrust bandwidth ranged from 5 to 1 rad/s. Tip-jet rotor speed bandwidth varied from 1.2 to 0.3 rad/s. CEPR bandwidth ranged from 5 to 1.2 rad/s. While hub pressure bandwidth ranged from 9 to 0.2 rad/s. The target bandwidth value of all the control parameters falls within the explored range.

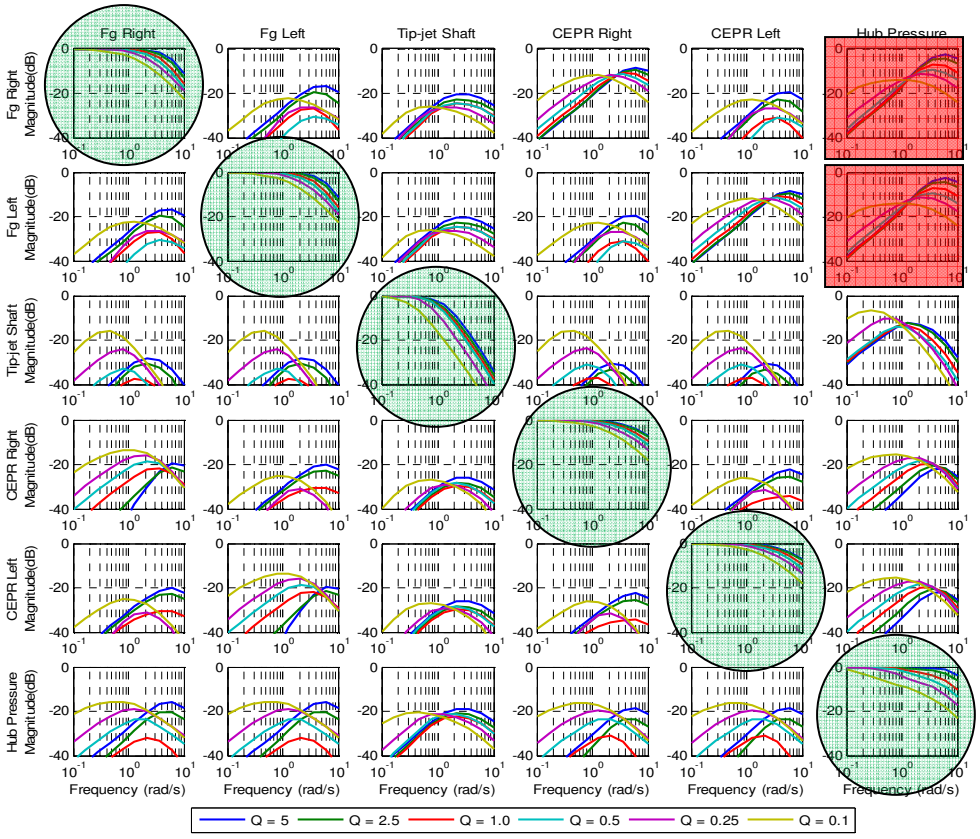

Figure 50: Bode Plot Magnitude Sensitivity to Changes in Reference Tracking Weighting Factor

As shown in Figure 50, for all design points tested, the interactions did not become significant. As with R, for high values of Q the high frequency interactions between the engine thrust and hub pressure demand become fairly noticeable. Similar to R, Q had a significant effect but opposite effect on the phase and gain margin. For all the control parameters, as the value of R decreases, both the phase and gain margin increased. The ranges seen in phase and gain margin were very similar to the ranges seen for R. In all these cases, the gain and phase margin exceeded the target values.

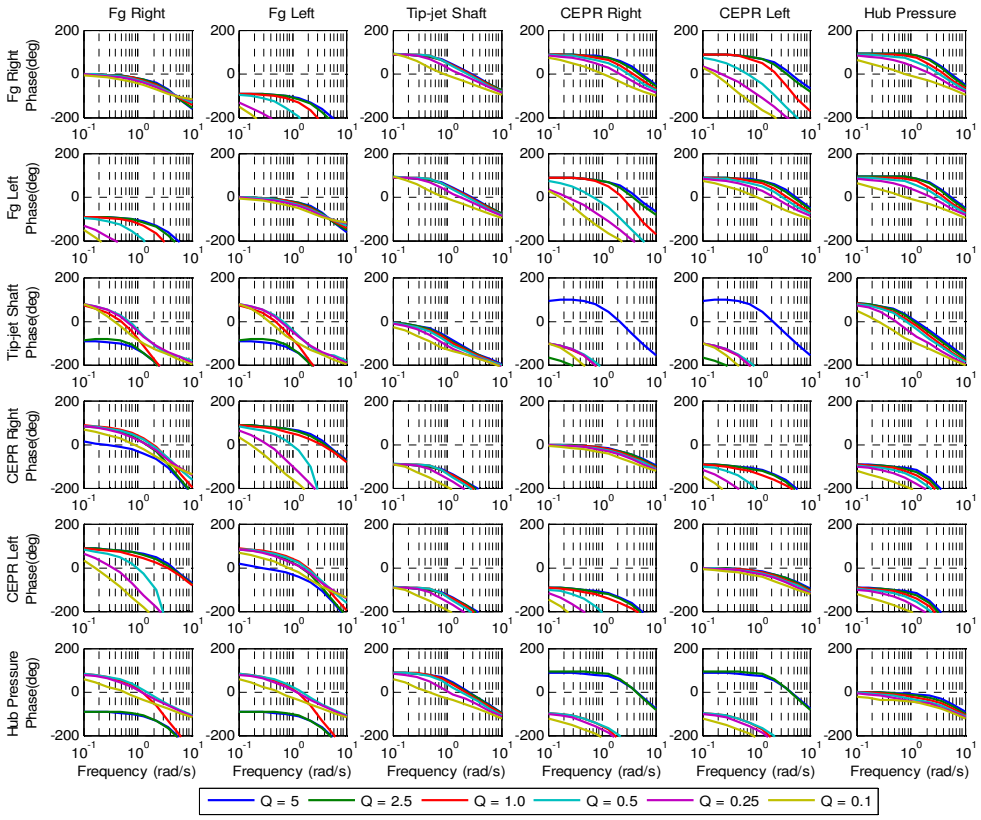

Figure 51: Bode Plot Phase Sensitivity to Changes in Reference Tracking Weighting Factor

7.3.1.3 Prediction and Control Horizon

Figure 52 and Figure 53 below show Bode plot sensitivity to changes in the control horizon. Given potential blocking schemes, as long as it is shorter than the prediction horizon, the control horizon can have many possible values. To keep the number of variables in this sensitivity study relatively small, no blocking scheme is explored. The control horizon ranged from equal the prediction horizon (H_u=100) to just a handful of time-steps (H_u=5). The control horizon did not have a significant effect on the bandwidth of the engine thrust or tip-jet rotor shaft speed. However, it did have a fairly significant effect on the bandwidth of CEPR and hub pressure. For both of these parameters, as control horizon decreased, bandwidth increased. The CEPR bandwidth ranged from 2.7 to 9 rad/s. However, the bandwidth increase is not linear with prediction

horizon and was only noticeable at very small prediction horizons. In fact, the control performance for all the parameters does not change much for the control horizon values ranging from 50 to 100. This is a very positive observation since the computational burden of the MPC is proportional to the control horizon.

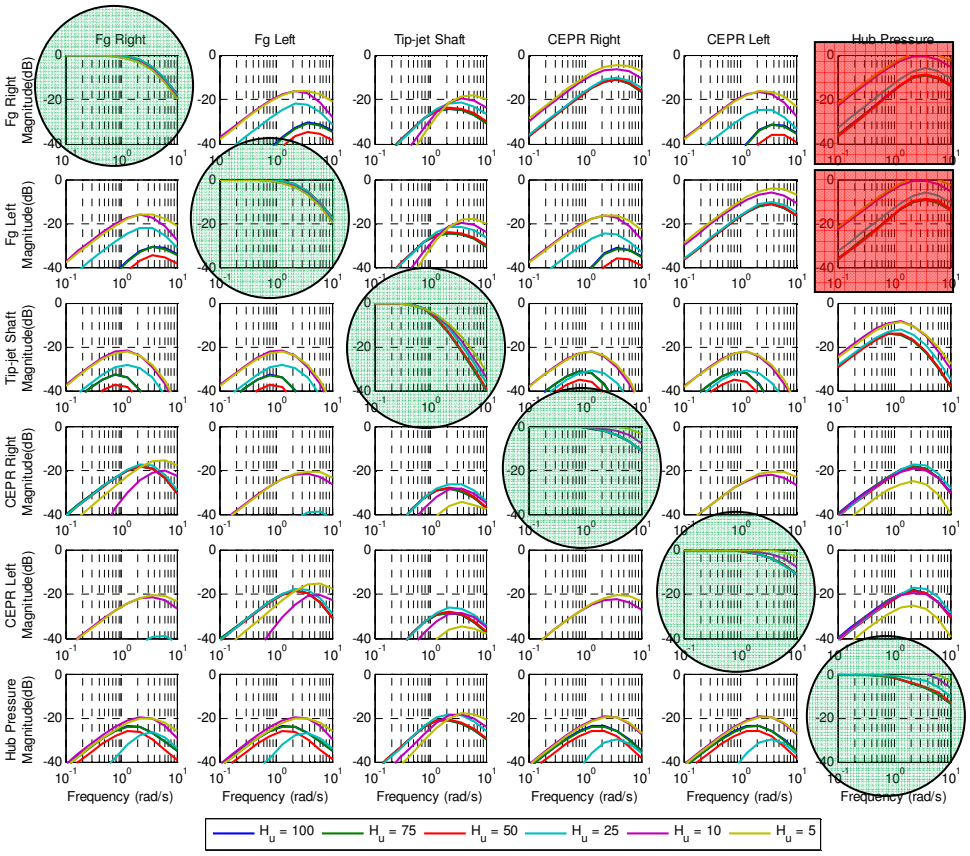

Figure 52: Bode Plot Magnitude Sensitivity to Changes in Control Horizon

As shown in Figure 52, at low values of H_u, the interactions between engine thrust and hub pressure demand at frequencies around 3 rad/s become significant. As with bandwidth, control horizon only affects the phase and gain margin of CEPR and hub pressure. As control horizon decreases, phase and gain margin decrease. CEPR ranged from 128° to 100° and 23 to 16 dB, respectively. Tip-jet rotor speed ranged from 140° to 90° and 25 to 15 dB, respectively. In all these cases, the gain and phase margin exceeded the target values.

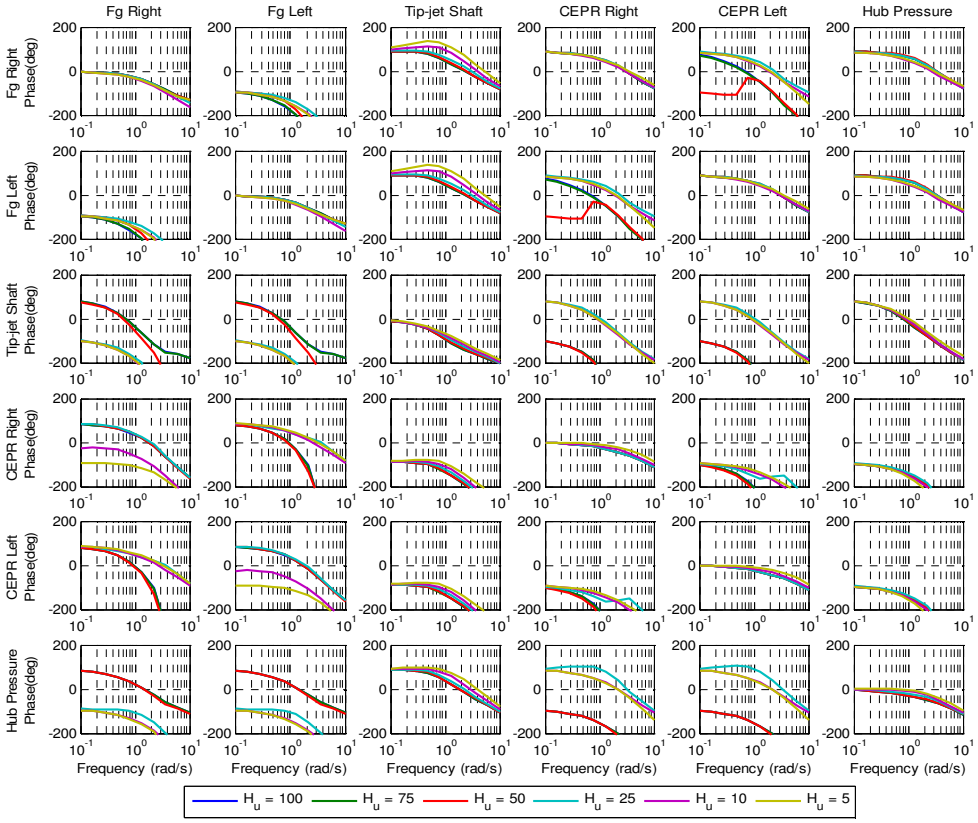

Figure 53: Bode Plot Phase Sensitivity to Changes in Control Horizon

Figure 54 and Figure 55 below show Bode plot sensitivity to changes in the prediction horizon. The prediction horizon ranged from 1.5 times larger than the tip-jet rotor time constant (H_p=150) to half the size of the engine shaft time constant (H_p=10). The prediction horizon did not have a significant effect on the bandwidth of the engine thrust or CEPR until the value of the prediction horizon approached the time constant of the engine shafts. Similarly, once the prediction horizon became smaller than the tip-jet rotor time constant, the bandwidth of the tip-jet rotor and hub pressure decreased.

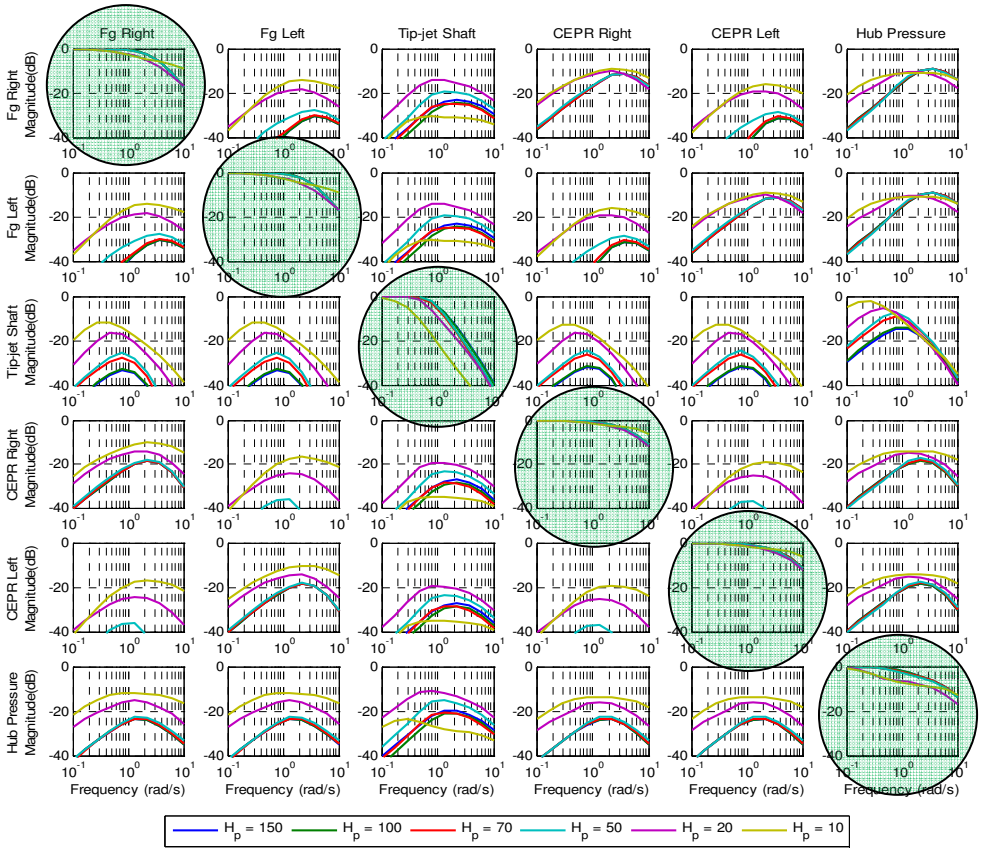

Figure 54: Bode Plot Magnitude Sensitivity to Changes in Prediction Horizon

As shown in Figure 54, for all design points tested, the interactions did not become significant. The trends seen in phase and gain margin are similar to that seen in bandwidth where degradation in margin only occurred when the prediction horizon became smaller than the dominant dynamic of the subsystem. In all these cases, the gain and phase margin exceeded the target values.

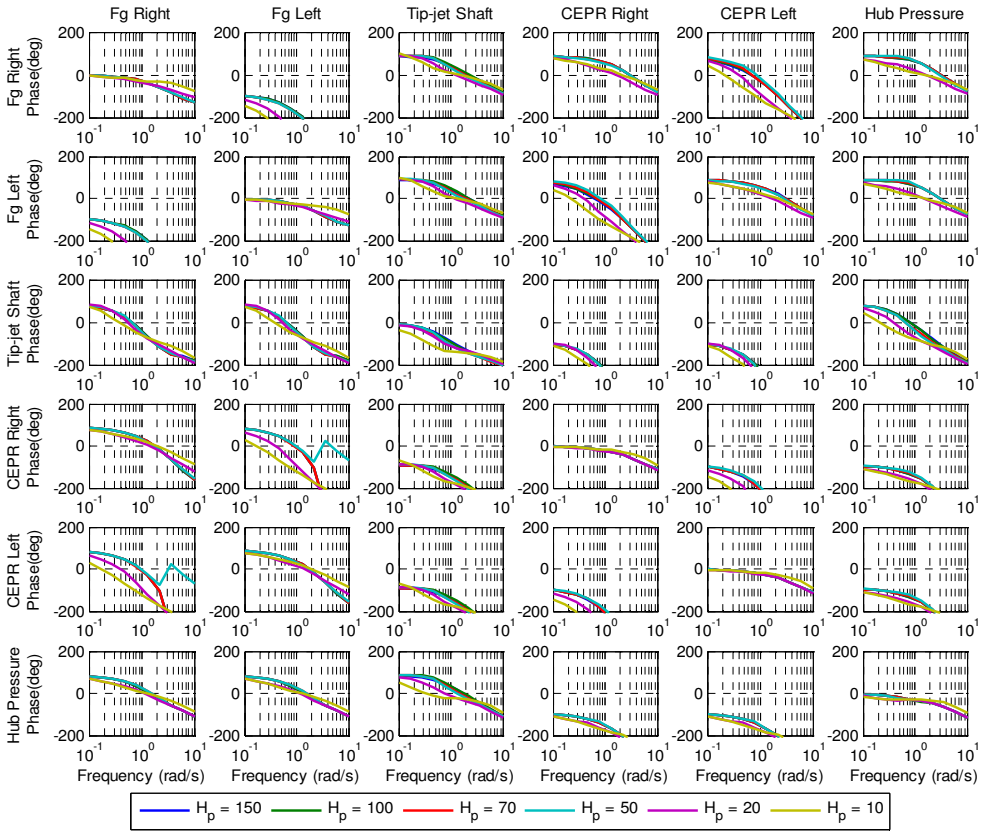

Figure 55: Bode Plot Phase Sensitivity to Changes in Prediction Horizon

7.3.1.4 Sensitivity Summary

From the above analysis, all the target bandwidths were attainable while maintaining enough phase and gain margin. However, it was noted that in general hub pressure was slower than thrust and CEPR. Table 21 summarizes the selection of the MPC design variables of the final MPC design variable choices. The prediction horizon was chosen to be 100. The cost function reference tracking weighting factor, Q, was chosen to be 0.5. The cost function control movement weighting factor, R, was 2.5. The reference tracking starting point for both the engine and tip-jet rotor were selected to be 15 and 90, respectively.

Table 21: Centralized MPC Candidates

Design Variable	Final Value
H_p	100
H_u	20
H_{we}	15
H_{wr}	90
Q	0.5
R	2.5

Table 22 below summarizes the performance metrics of the final centralized MPC design variable selections. In general, all the bandwidth targets are near the targets except for hub pressure. This fact may be acceptable for MPC because of the way MPC handles constraints. Since hub pressure and CEPR are both primarily correlated to stall margin, they can be considered safety related control parameters; whereas thrust and tip-jet rotor shaft speed are related the performance of the overall system. From the perspective of the vehicle using the tip-jet reaction drive system, as long as thrust and tip-jet rotor shaft respond as desired, it does not care that the pressure response is slower than desired given that none of the constraints are violated. Thus CEPR and hub pressure control can be viewed as more of a mechanism to drive the system to the desired steady-state performance, while using the MPC to ensure that no constraints are violated during a transient.

Table 22: Centralized MPC Performance Metrics

Centralized MPC Control Performance	Bandwidth rad/s	Phase Margin degrees	Gain Margin dB
Fg Right	2.46	114	49
Fg Left	2.46	114	49
Tip-jet Shaft	0.8	101	28
CEPR Right	2.78	128	23
CEPR Left	2.78	128	23
Hub Pressure	1.61	140	25

7.3.2 Sensitivity to Modeling Error

With the integration of a model in model based control methods, the sensitivity of the control to modeling errors must be accounted for. As mentioned previously, sources of modeling error can come from various aspects of the onboard model modeled wrong or the actual system performing differently than what is captured in the onboard model.

To capture the effect of modeling error on the candidates of centralized MPC for the tip-jet reaction drive system, four different modeling error simulations were performed: turbine map error (both efficiency and flow), fan map error (both efficiency and flow), reaction drive duct pressure loss error, and system degradation. For all these simulations, the onboard model is calibrated to the actual system using data matching scalars. To model the turbine map error case, both the turbine efficiency and flow map were scaled by 3%. Similar to turbine modeling error case, the fan map error was modeling by scaling the fan flow and efficiency maps by 3%. The reaction drive duct pressure loss was simulated by scaling the duct pressure loss by 5%. The degraded system case was modeled by scaling the fan, HP compressor (HPC), HP turbine (HPT), and LP turbine (LPT) efficiency and flow maps by factors expected after a significant operational period. The metrics used to compare the candidates was the percent difference between the actual and onboard model estimated non-measured parameters and the percent difference between the actual and target performance metrics. The better candidate should minimize the error between onboard model and the actual system as well as operate at the target parameters.

Table 23 summarizes the sensitivity to modeling error of the final centralized MPC design selection. The first case to be analyzed was the turbine map modeling error. For this case, the onboard model overestimates actual engine thrust and hub pressure by about 2.5% and 0.24%, respectively. LP and HP stall margin are overestimated by the model by 8% and 18%, respectively. This fact may result in margining of the stall

margin constraint that is used in the control. Turbine inlet temperature is underestimated by about 2.5%.

The onboard model matches the actual system performance much better for the fan map error case. The main reason for this is because both the flow and efficiency map discrepancy for the fan, as opposed to the turbine, can be fully observed using the available sensors and the component data match scalars. All the mode estimated safety and performance parameters match the actual parameters within 0.5% with virtually no error in thrust.

Table 23: Centralized MPC Sensitivity to Modeling Error

	Sensitivity to Modeling Error	Turbine Map Error	Fan Map Error	Reaction Drive Pressure Loss Error	Degraded System
Model-Actual % Difference	Thrust	2.37%	0.00%	-0.60%	0.27%
	LP Stall Margin	7.71%	0.20%	4.86%	2.30%
	HP Stall Margin	17.79%	0.50%	0.50%	5.20%
	Turbine Inlet Temperature	-2.60%	-0.10%	0.44%	-1.80%
	Hub Pressure	0.24%	0.00%	-1.70%	0.92%
Target-Actual % Difference	Thrust	2.37%	0.00%	-0.60%	0.27%
	CEPR	0.00%	0.00%	0.00%	0.00%
	Ntip	0.00%	0.00%	0.00%	0.11%
	Hub Pressure	0.24%	0.00%	-1.70%	0.92%
Data Match Scalars	Fan Eta	1.002	0.969	0.999	0.980
	Fan Wc	0.994	0.971	1.040	0.980
	HPC Eta	0.992	1.000	0.999	0.979
	HPC Wc	0.989	1.000	0.998	0.978
	HPT Wc	1.003	1.000	1.000	1.013

Similarly to the turbine fault, when the reaction drive duct pressure loss model is wrong, the model estimated parameters differ from the actual parameters. However the main differences appear in the LP stall margin and hub pressure estimates which vary by

5% and 1.7%, respectively. When the pressure loss in the duct is underestimated, the actual hub pressure is higher than the estimated pressure resulting in a significantly lower steady state fan stall margin. This fact makes highlights the importance of having a good rotor pressure loss model.

For the degradation case, the onboard model matches actual thrust by within 0.5%. However, both stall margin and turbine inlet temperature differ from the actual system by at least 2%. In this case the hub pressure is over estimated by around 1% resulting in an increase in steady state fan stall margin.

7.3.3 Disturbance Rejection

In addition to good control input to control output response, it is important the controlled parameters do not deviate significantly from the demanded values when faced with a change in rotor load. Below in Figure 56 the responses of the MPC to various different sinusoidal inputs of the rotor load. Besides the tip-jet rotor shaft speed, none of the controlled parameters is affected by a change in rotor load. Similarly to the PI controller in Figure 21, the maximum response by the tip-jet rotor speed of about -7 dB occurs around the bandwidth frequency. Although not insignificant, the magnitude of the response is less than the -3 dB defined as the criteria for determining control bandwidth. At input frequencies greater then around 3 rad/s tip-jet rotor speed is not affected at all by changes in rotor load. Therefore from this analysis, the control system should be able to suitable reject changes in the rotor load.

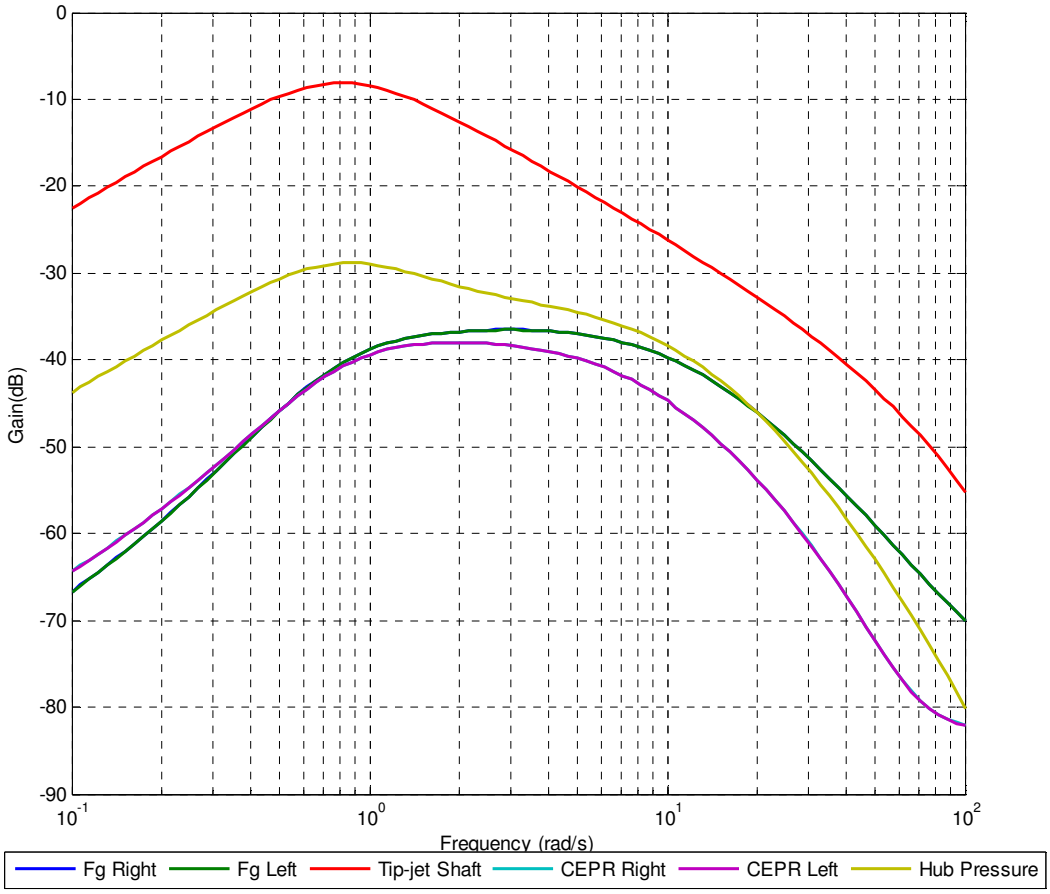

Figure 56: Centralized MPC Response to Rotor Load Disturbance

7.3.4 Design Summary

Besides hub pressure, all the control parameters were near their target values. The integration of an optimization algorithm may help in the fine tuning of the candidates to better meet the performance metrics. Although a control horizon value of 50 was chosen for the final design, a design that met all the performance metrics with a minimal length in the control horizon would be most optimal. The sensitivity to modeling error study can used to select the appropriate onboard model used in the controller (Real-Time Modeling Methods for Gas Turbine Engine Performance July 2001)(SANGHI, LAKSHMANAN and SUNDARARAJAN 2000). Lastly it was shown the rotor load disturbance properties of the controller were acceptable. From here different time based

simulations of the centralized MPC can be performed to analyze the controller performance.

7.4 Centralized MPC Simulations

The purpose of the simulations in the next few sections is to provide an understanding of the overall performance/response of the centralized MPC. Specific design variables such as varying throttle settings, disturbance rejection, limit avoidance, and robustness to degradation are explored in detail.

7.4.1 Rotor Load and Throttle Demand Change Simulations

The following sections are broken up into analysis of engine throttle changes, rotor load demand changes, and both engine throttle and rotor load demand changes. The engine throttle changes provide an opportunity to analyze how the control responds when multiple demands are changed simultaneously. For a tip-jet reaction drive system, a change in rotor load demand provides a natural opportunity to analyze disturbance rejection. And the final section will combine the two types of demand changes to see if there are any additional observations.

7.4.1.1 Throttle Demand Changes

The first transient to analyze is a change in engine throttle demand. As the engine throttle demand changes, the engine thrust demand changes. The CEPR demand also changes with throttle demand to maintain close to design compressor stall margin. Although physically associated with the tip-jet subsystem, the hub pressure demand changes with throttle changes because of its strong correlation with fan stall margin. The demand schedules were defined and developed while the cycle was operating at new and clean conditions (i.e., component performance matched component map performance).

Figure 57 and Figure 58 below shows a step change in throttle demand for both a new and clean system. In all the transients, it takes the approximately two seconds for the engine thrust and hub pressure to settle to the new demand values. CEPR has some overshoot in both transients. This effect is from the interaction with thrust demand. In non model predictive control architectures, this aspect may be quite negative because of the resulting reduction in stall margin during some transients; but model predictive control can ensure that the minimum required stall margin is never encountered. Additionally since hub pressure and CEPR are not vehicle performance parameters and the overshoot is not seen in the performance related parameters, this overshoot may be acceptable.

From the perspective of the tip-jet subsystem, the change in engine throttle has a two-fold effect. Firstly, the tip-jet rotor speed demand remains constant while the demanded hub pressure changes. This is in contrast to the engine subsystem where both thrust and CEPR demands change with respect to a throttle change. Secondly as the engine thrust changes, the engine speed changes result in a reduction in the mass flow into the tip-jet subsystem. For the PI controller, this interaction is viewed as an unmeasured disturbance which the control needs to reject the amount of time required to reject the disturbance is limited by the time constant of the tip-jet rotor speed controller. However, for centralized MPC, it is captured in the onboard model and is a known interaction. However this effect is book kept, the general effect is the same, which is a change in the tip-jet rotor speed. To overcome this effect, the tip-jet fuel flow is increased to generate more torque (as seen through an increase in velocity) in the tip-jet rotor. In contrast to the PI controller, the centralized MPC takes only 5 seconds for the tip-jet shaft speed to reach steady state. Unlike the PI controller, the disturbance in the tip-jet shaft speed is almost unnoticeable. This is a very positive improvement because unlike the distributed PI controller, the tip-jet shaft speed is not noticeable affected by changes in the engine.

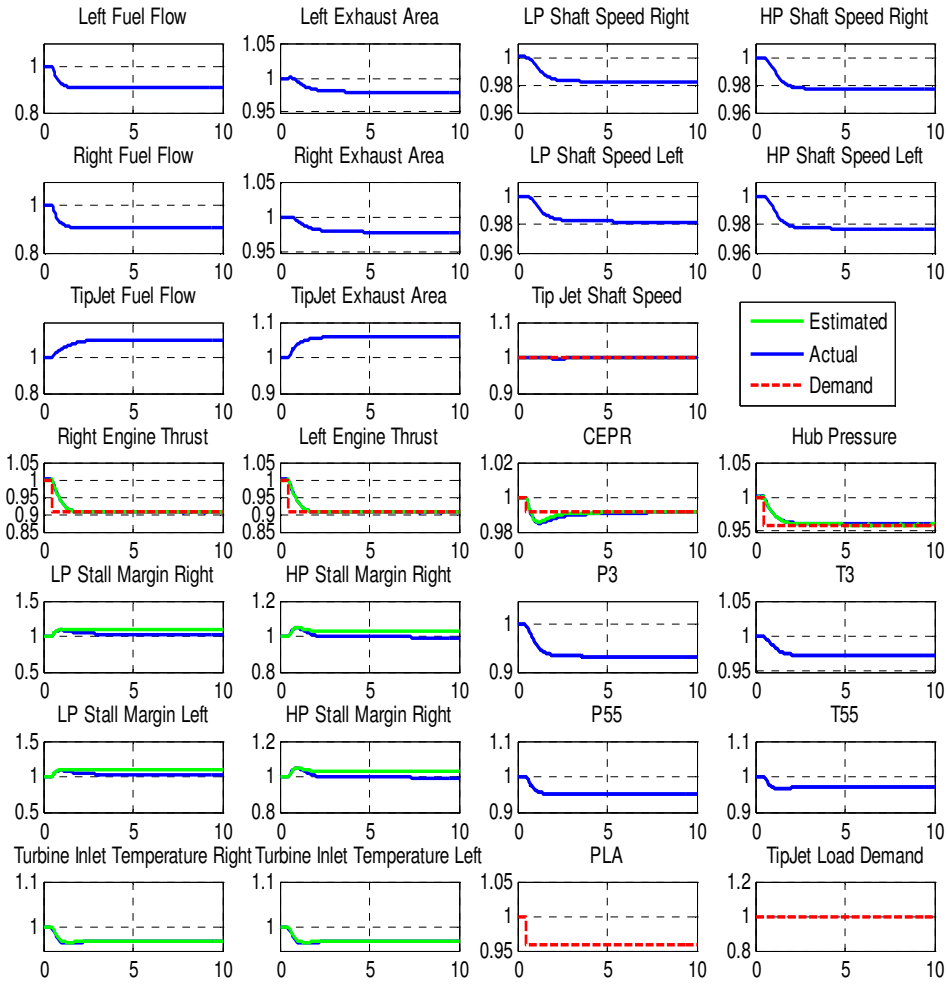

Figure 57: Engine Throttle Demand Decrease for a New and Clean System

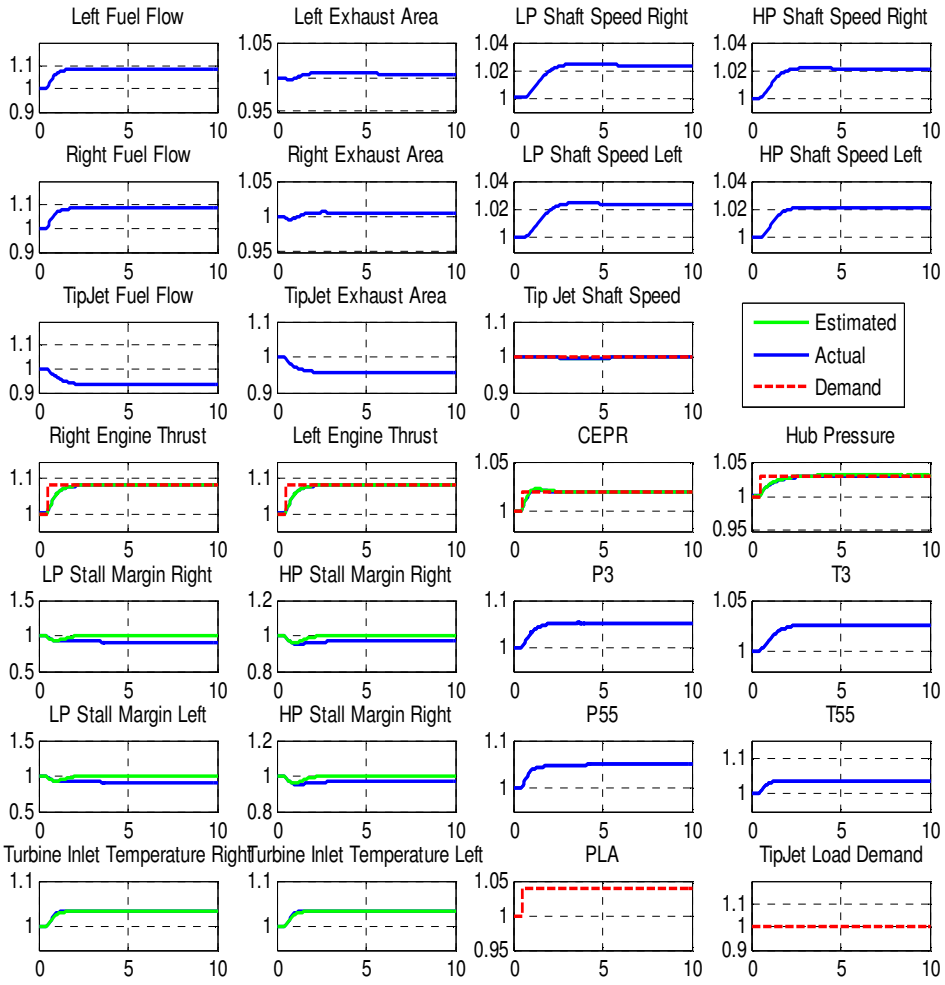

Figure 58: Engine Throttle Demand Increase for a New and Clean System

Figure 59 and Figure 60 below show a step change in throttle demand for a degraded system. As expected the temperatures in for the degraded system are elevated relative to their new and clean counterparts. Besides the elevated temperatures, the response of the degraded system is virtually identical to that of the new and clean system.

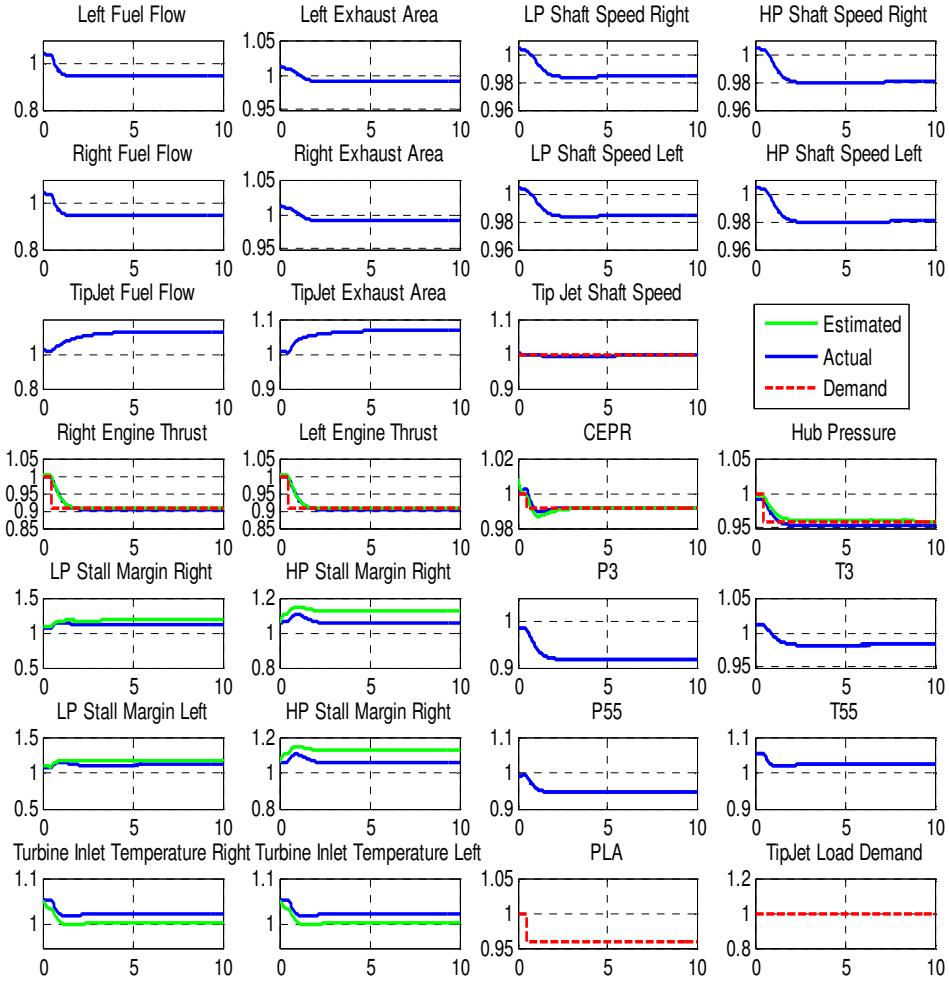

Figure 59: Engine Throttle Demand Decrease for a Degraded System

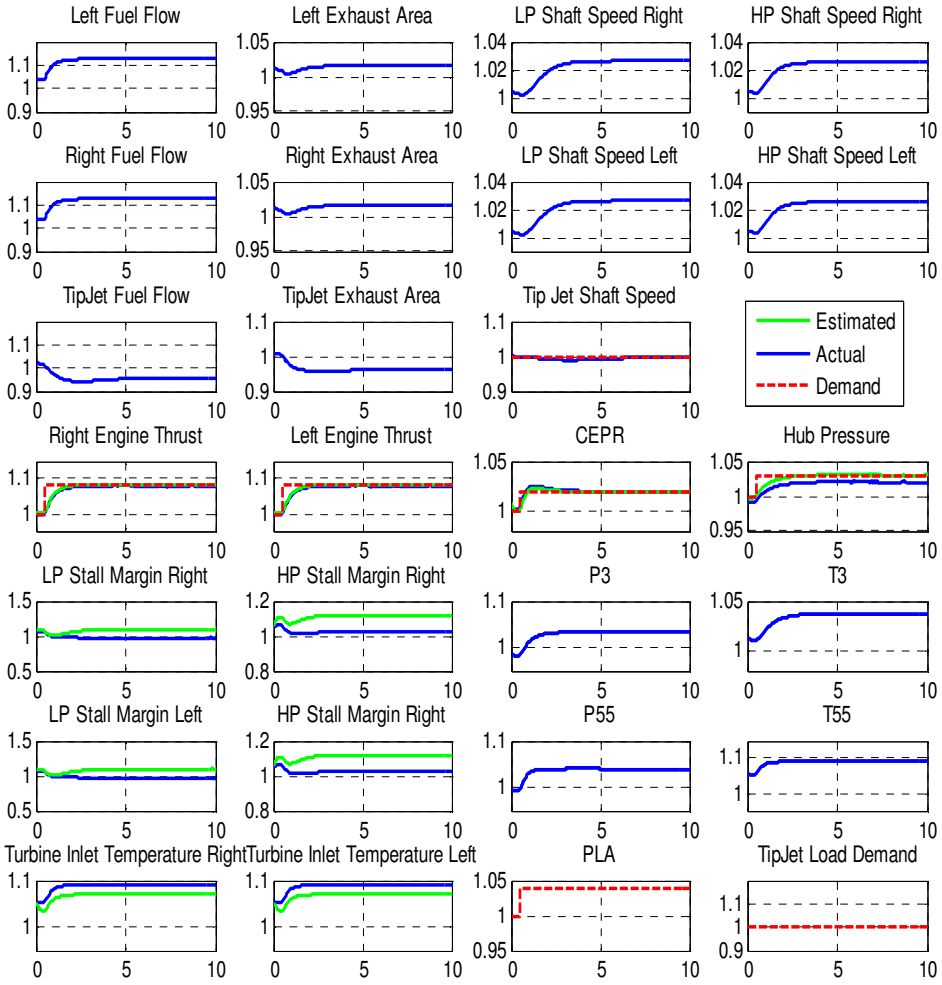

Figure 60: Engine Throttle Demand Increase for a Degraded System

Throughout all the transients there is a noticeable difference between the stall margins estimated by the onboard model and the actual stall margin. In general it appears that the onboard model typically overestimates the available stall margin. To overcome this limitation, the minimum amount of stall margin used in the constraint may need to be margined to ensure the system doesn't stall because of model uncertainty.

For the new and clean system, the onboard model provides a fairly accurate estimate of the turbine inlet temperature. However as the system degrades, the onboard

model underestimates turbine inlet temperature by about 1%. To account for this in the constraints, the constraints should be margined by about 1% to ensure the actual turbine inlet temperature does not exceed any design values, thus reducing the available life of the system.

7.4.1.2 Rotor Load Demand Changes

Another transient of interest for the tip-jet reaction drive system is the response to a change in tip-jet rotor load. This load change can be induced from either direct pilot input or indirectly from changes in ambient conditions, such as a wind gust. From the perspective of the tip-jet subsystem control, or more generally the overall control, a rotor load change is viewed as a disturbance. Since it is a disturbance, none of the speed or pressure demands change when a rotor load change is encountered. As compared with the PI controller, the centralized MPC views changes in the rotor load as measured disturbances, where the effect of changes in the load on the system is known.

Figure 61, Figure 62, Figure 63, and Figure 64 below show rotor load increase and decreases on both new and clean and degraded systems. Similarly to the throttle change simulations, the system response for all four scenarios is very similar. For all the transients, the tip-jet rotor shaft speed takes around 5 seconds to settle after exposed to a 10% change in rotor load. With the only significant differences between the new and clean and degraded system cases are the elevated speeds and temperatures of the degraded system. This is similar to the settling time seen for the tip-jet shaft for the throttle change simulations. Slight perturbations in CEPR and hub pressure can be seen. Although the order of magnitude of these are much smaller when compared with effect of the disturbance on the tip-jet shaft speed. There does not appear to be any significant effect of the engine thrust to changes in the rotor load.

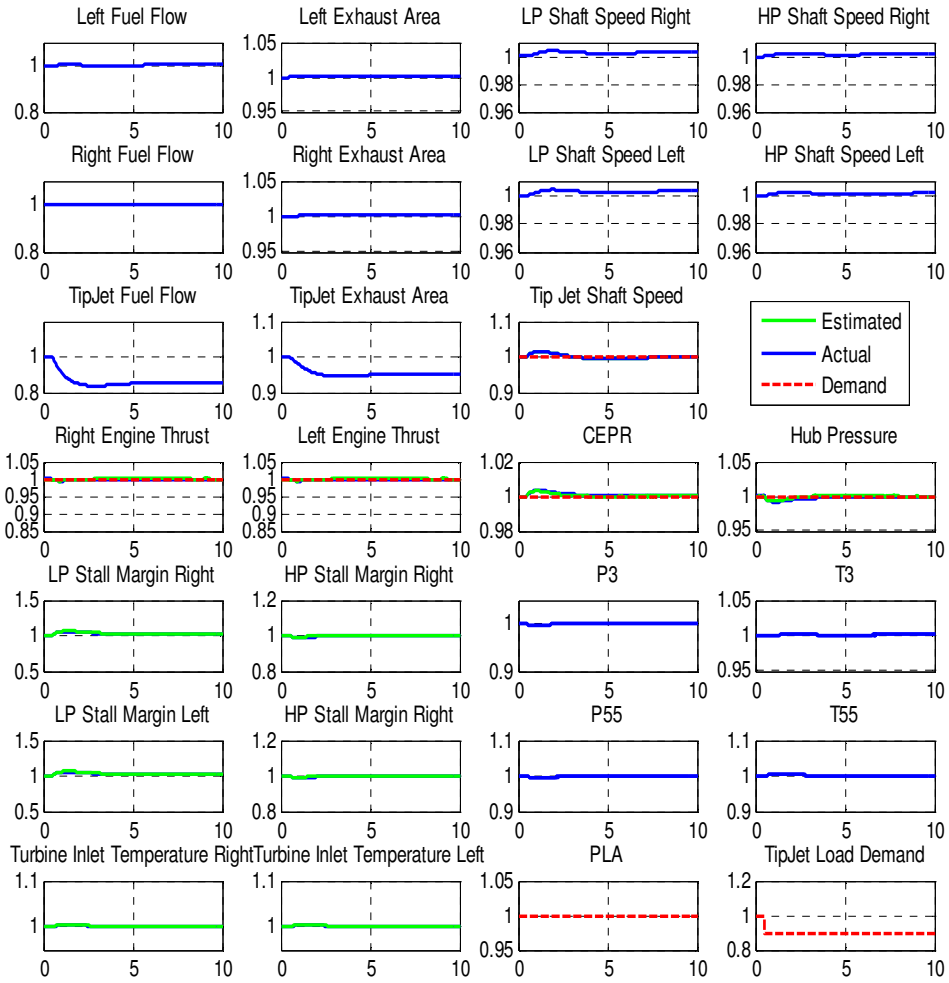

Figure 61: Rotor Load Decrease for a New and Clean System

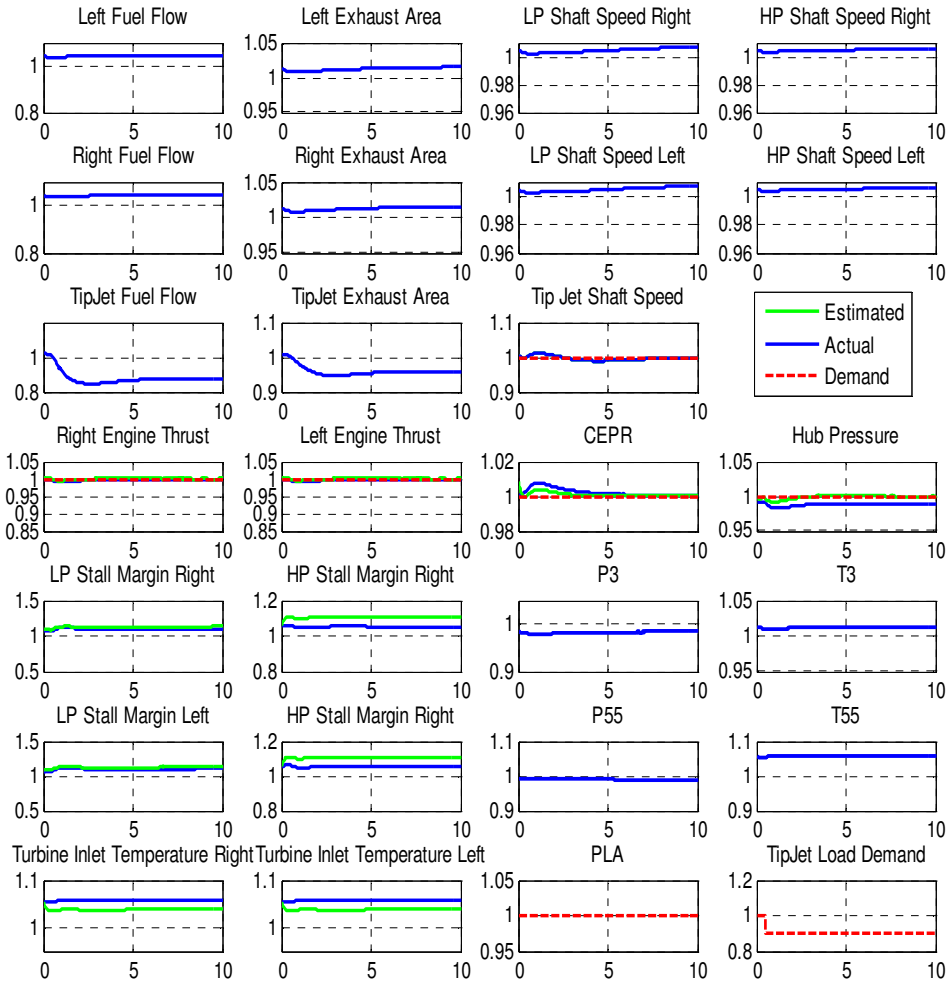

Figure 62: Rotor Load Decrease for a Degraded System

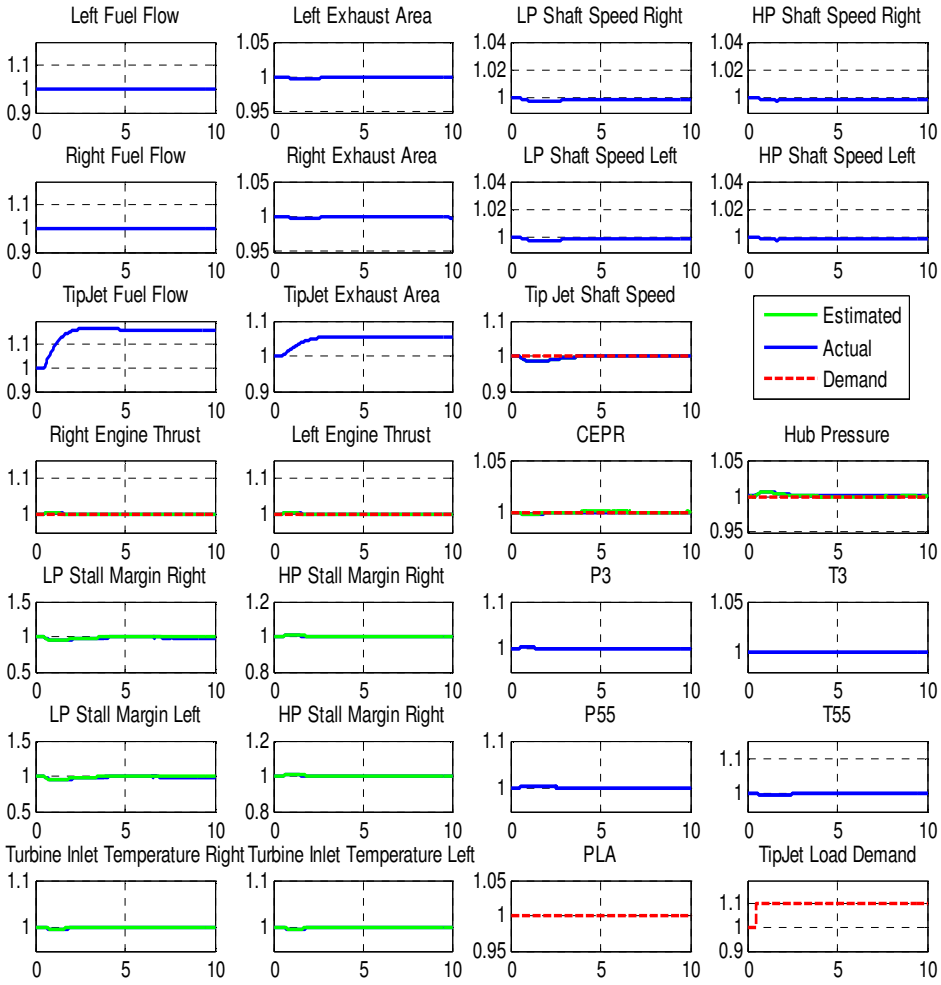

Figure 63: Rotor Load Increase for a New and Clean System

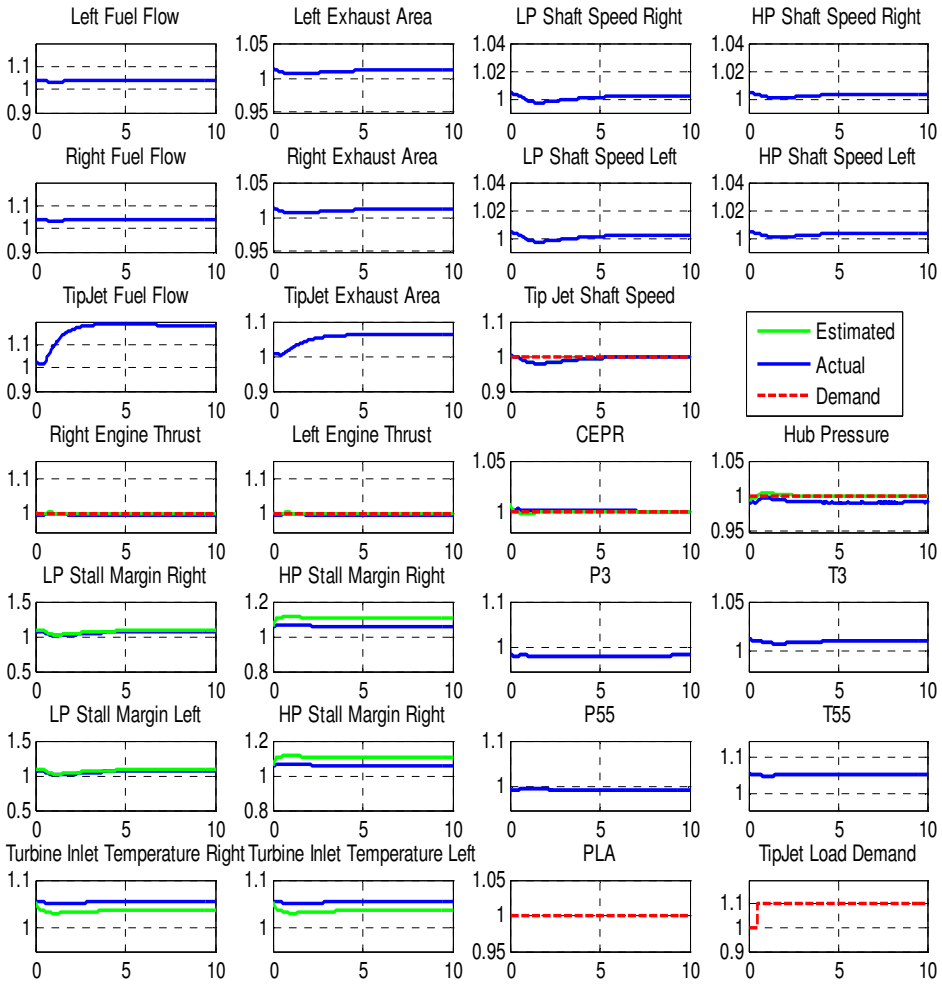

Figure 64: Rotor Load Increase for a Degraded System

7.4.1.3 Throttle and Rotor Load Changes

The last transient to analyze combines the two previous transients. Figure 65, Figure 66, Figure 67, and Figure 68 below show rotor load/engine throttle increases and decreases on both new and clean and degraded systems. Matching the previous MPC simulations, the tip-jet rotor speed settles approximately 5 seconds after the initialization. And the other demanded parameters settle on the order of 2 seconds after the transient starts. Again CEPR has the overshoots seen in the previous simulations. In the general the trends in these simulations are similar to the trends seen in the previous runs. There

do not appear to be any additional interactions or effects resulting from the combination of the two demand changes; and the conclusions drawn from the previous sections can be applied to the analysis of these transients.

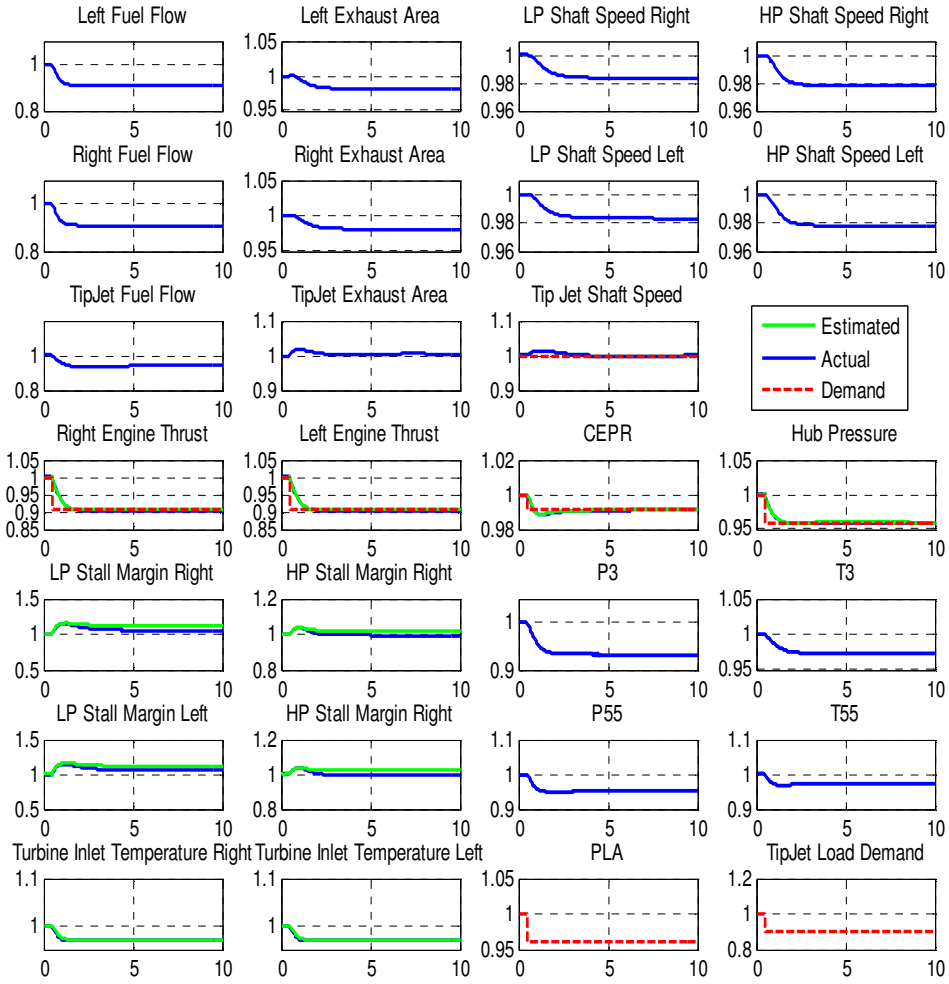

Figure 65: Engine Throttle and Rotor Load Decrease for a New and Clean System

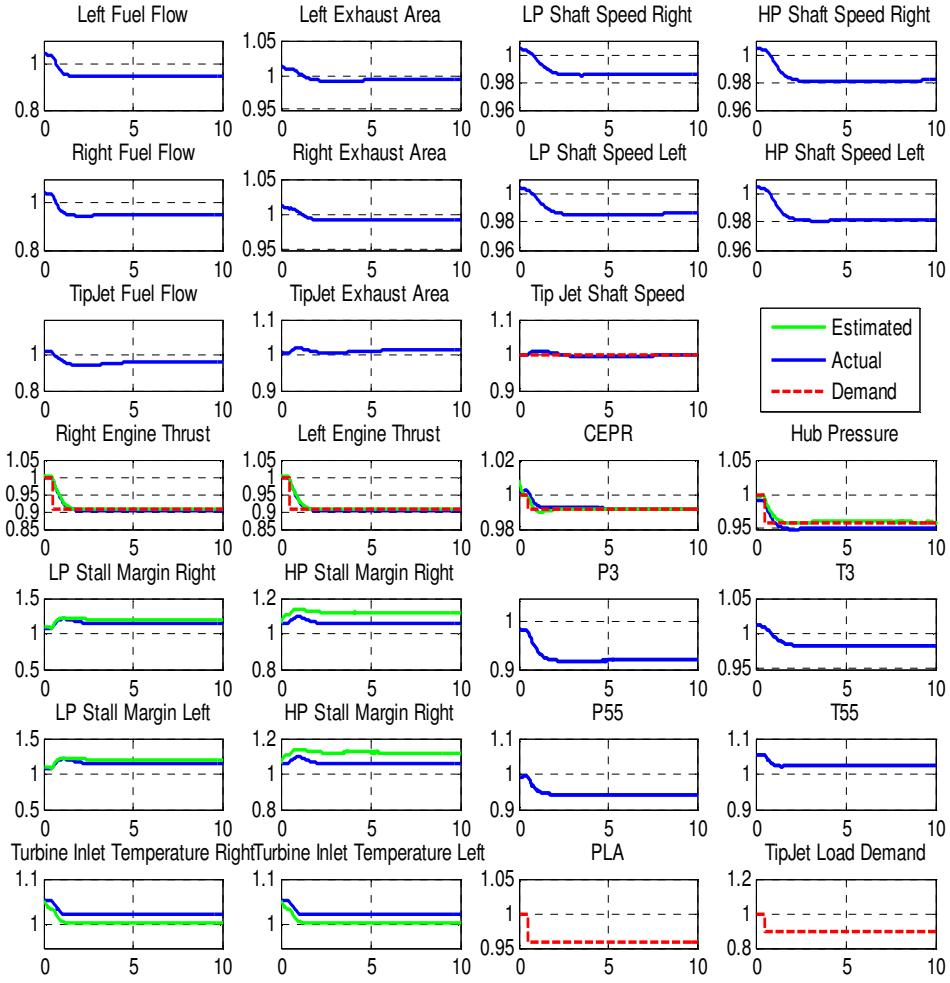

Figure 66: Engine Throttle and Rotor Load Decrease for a Degraded System

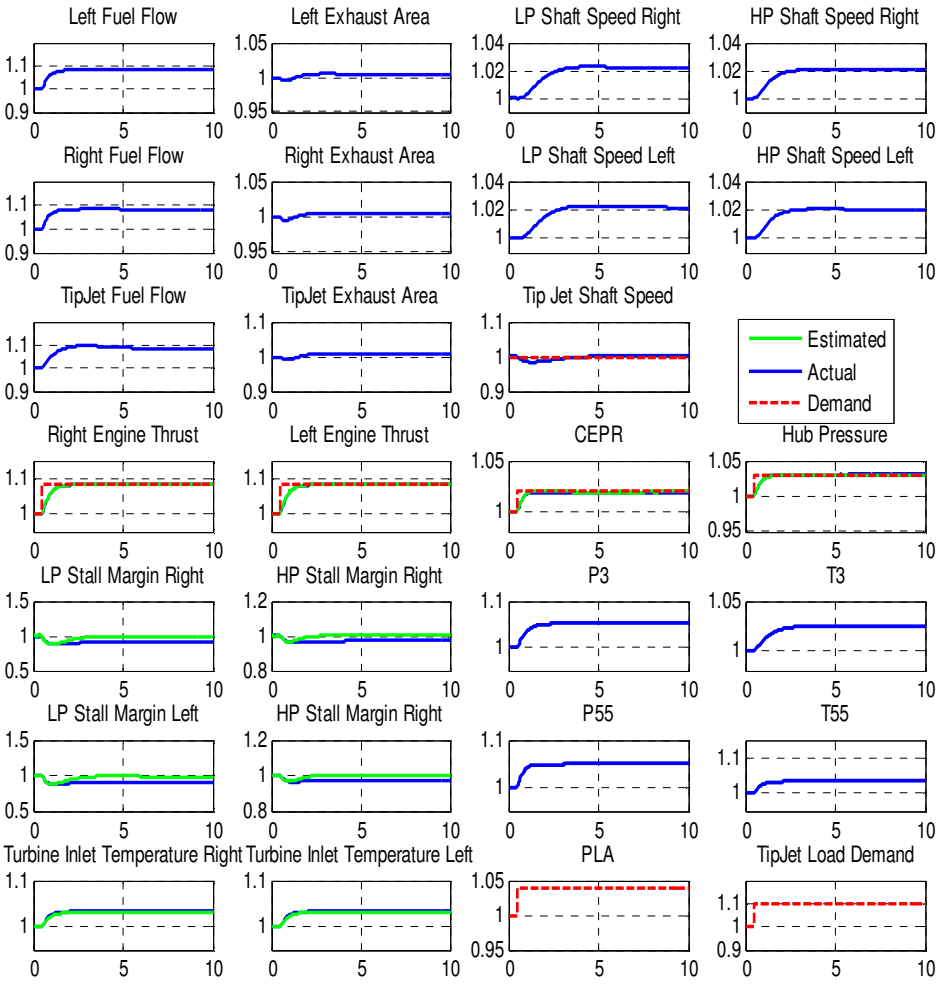

Figure 67: Engine Throttle and Rotor Load Increase for a New and Clean System

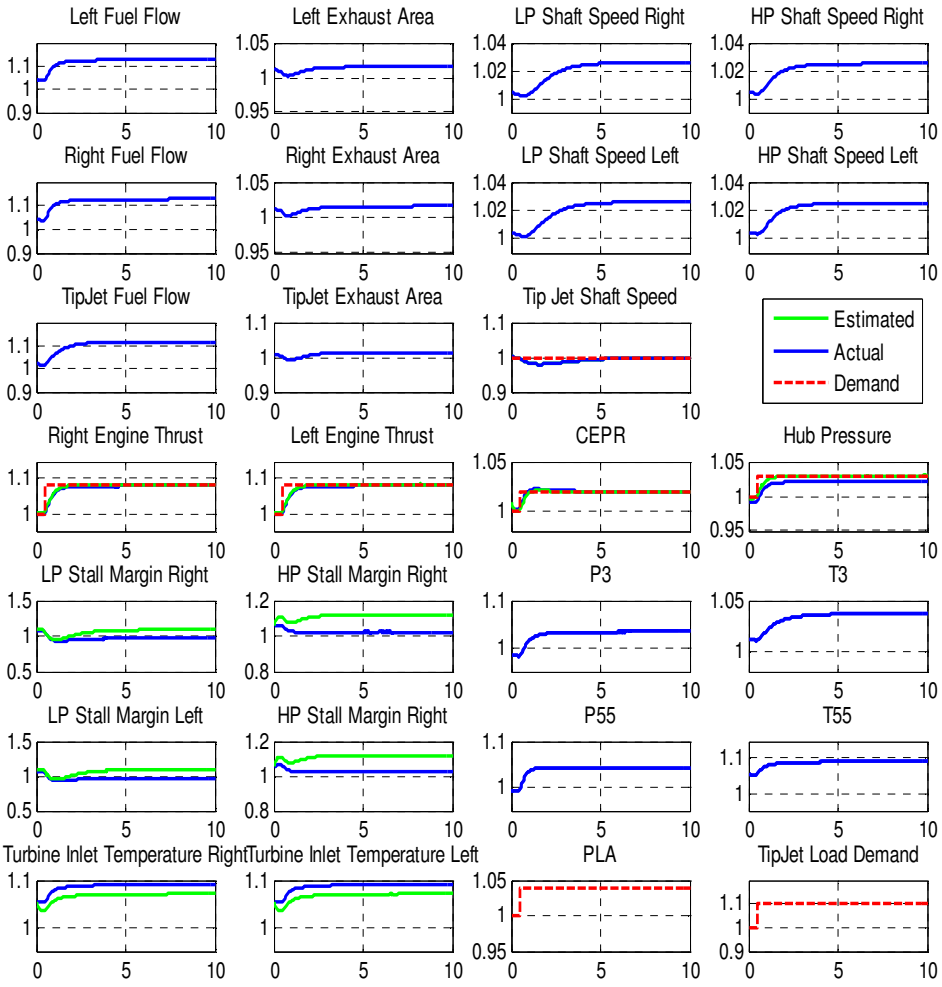

Figure 68: Engine Throttle and Rotor Load Increase for a Degraded System

7.4.2 Constraint Handling

As has been mentioned throughout, a primary benefit of MPC is the handling of constraints. The use of constrained optimization techniques allow the integration of constraints handling and reference tracking into a single control framework. MPC can handle constraints on measured output, unmeasured output, control input position, and control input rate change. To demonstrate centralized MPC constraint handling, two

cases of interest will be explored: stall margin limit avoidance and control input rate limitations.

7.4.2.1 Engine Stall Margin

To demonstrate the centralized MPC stall handing capability a simulations were run where the hub pressure demand increased while holding all other demands constants. In this case the fan pressure ratio will increase resulting in the system operating at a lower stall margin steady-state fan stall margin. Figure 69 and Figure 70 below shows plots of both the unconstrained and constrained simulations. The former is unconstrained and the latter has a constraint on both the LP and HP stall margin. For the purpose of these simulations, the stall margin was constrained to be no lower than 75% of the design value. In reality this value would be much lower, however to ensure the nonlinear system model converges in the unconstrained case, the stall margin limit was defined at such a high level. In the constrained simulation, the LP stall margin constraint becomes active resulting in the system operating with a lower than demanded hub pressure.

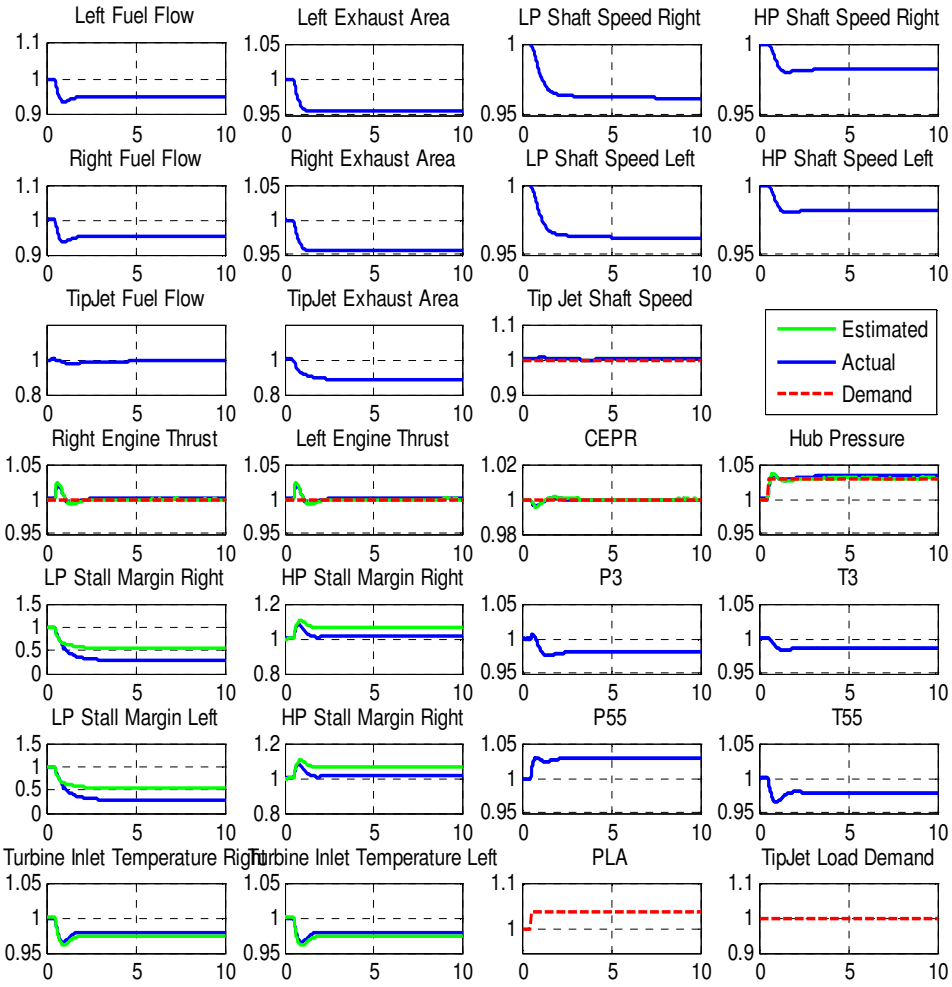

Figure 69: Unconstrained Hub Pressure Increase

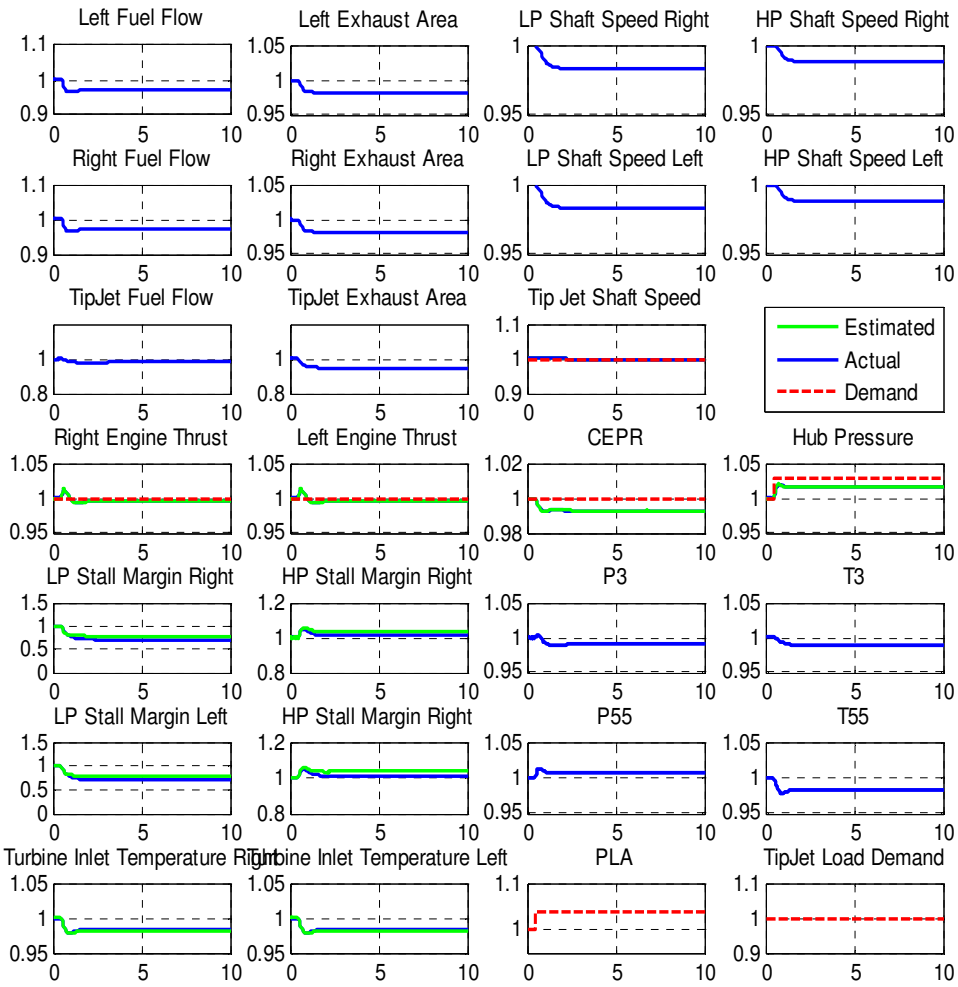

Figure 70: Stall Margin Constrained Hub Pressure Increase

Figure 71 below compares some of the control parameters for both the constrained and unconstrained cases. Hub pressure which is most directly correlated with stall margin is significantly reduced when the LP stall constraint is active. This is quite positive because although the system operator demands a potentially unsafe condition, the control will not allow the tip-jet reaction drive system to operate unsafely.

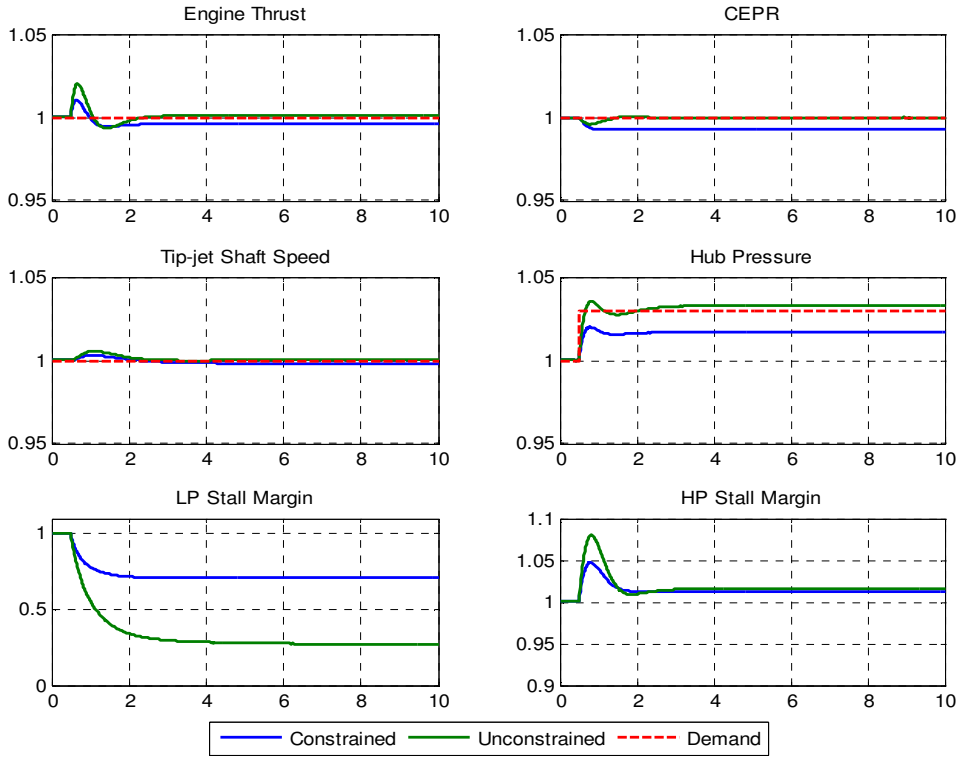

Figure 71: Centralized MPC Control Performance for Stall Margin Limit Avoidance

7.4.2.2 Actuator Rate Limits

Figure 72 and Figure 73 show two rotor load decrease simulations. The former is unconstrained and the latter has a constraint on actuator rate of both the tip-jet fuel flow and exhaust area. The values of the rate limits for both actuators are the same. The rate limits were chosen such the one of the limits would be active during the transient. For the constrained case, the tip-jet fuel flow actuator is operating at its maximum rate of change from the beginning of the transient till almost 6 seconds afterwards. Looking at these plots it is obvious that the tip-jet shaft speed responds much slower when the actuator rate limits are active.

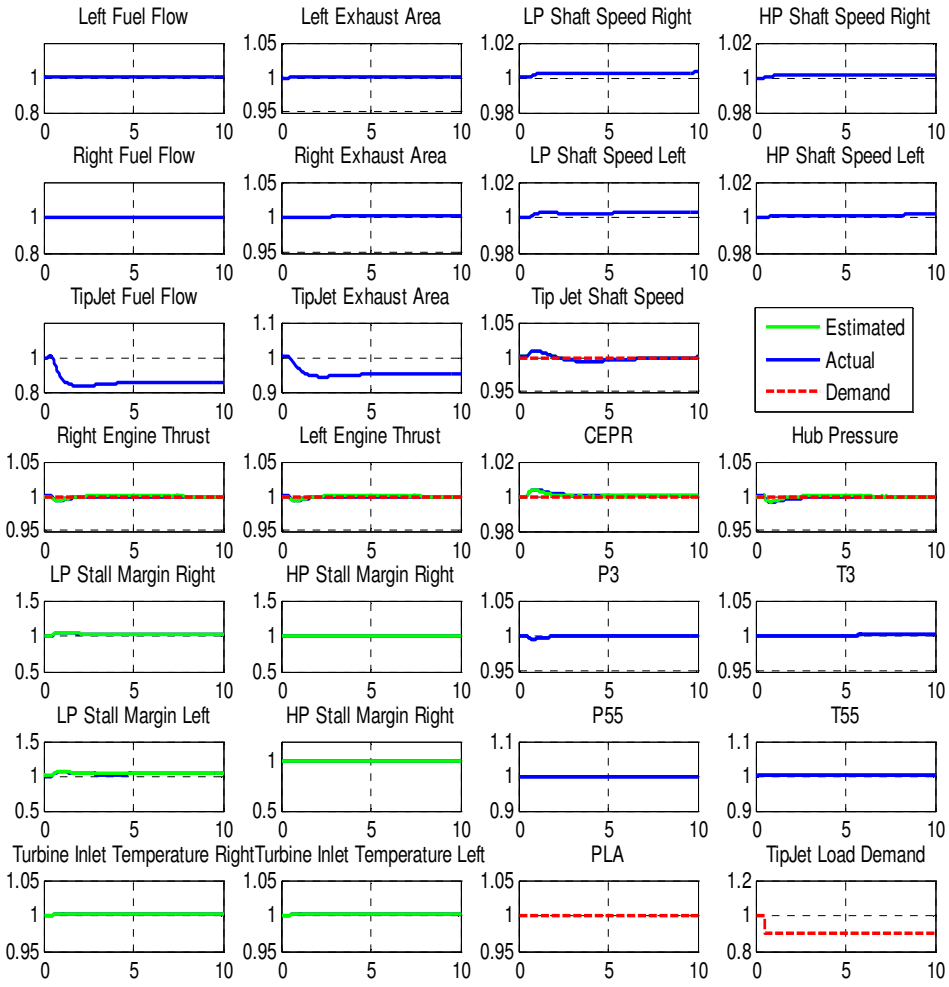

Figure 72: Unconstrained Rotor Load Decrease

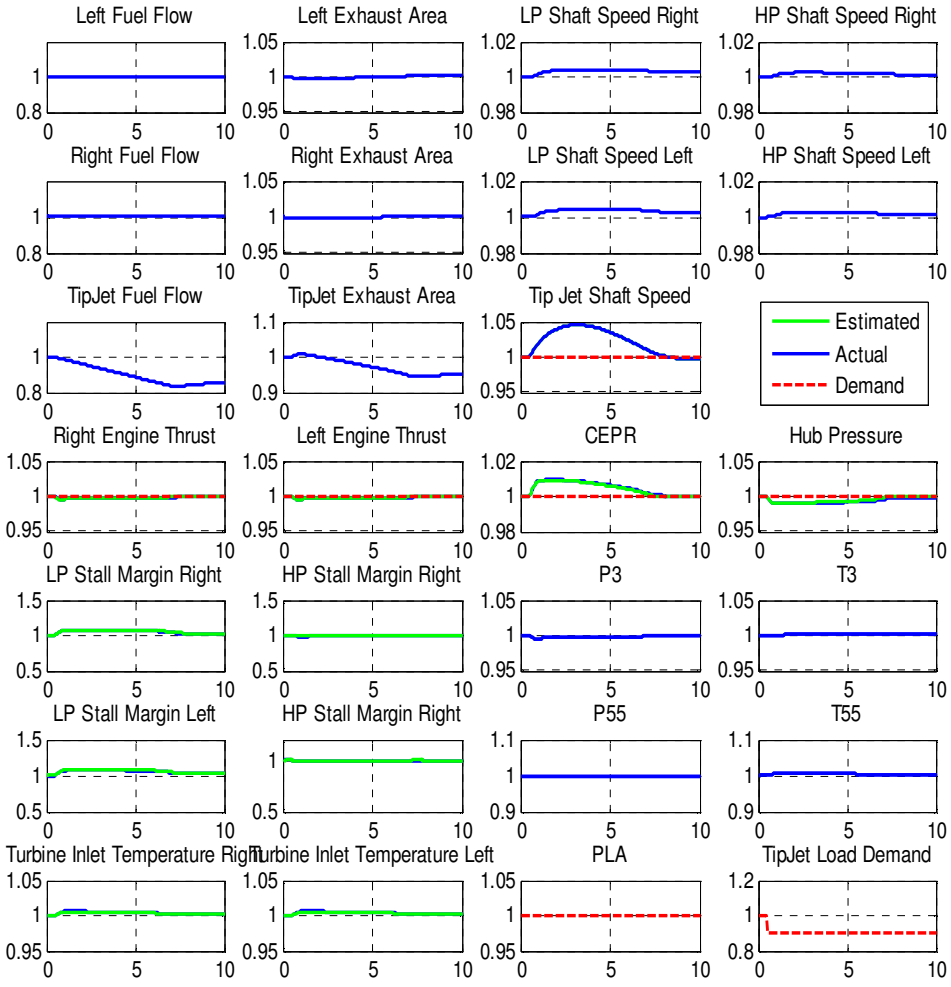

Figure 73: Actuator Rate Limit Constrained Rotor Load Decrease

Figure 74 below compares some of the control parameters for both the constrained and unconstrained cases. For the constrained case, the tip-jet rotor speed takes about 2 seconds longer to settle than the unconstrained case with a much more noticeable disturbance in tip-jet shaft speed. This is a direct result of the tip-jet fuel flow being rate constrained. At first glance it appears that tip-jet exhaust area is also constrained. However the rate of change of the tip-jet exhaust area is slower than the tip-jet fuel flow. Since the fuel flow is larger for the constrained case, the tip-jet exhaust

area needs to be more open to maintain target hub pressure. Hub pressure for the constrained case is lower than target. To maintain close to target thrust, the CEPR is correspondingly increased.

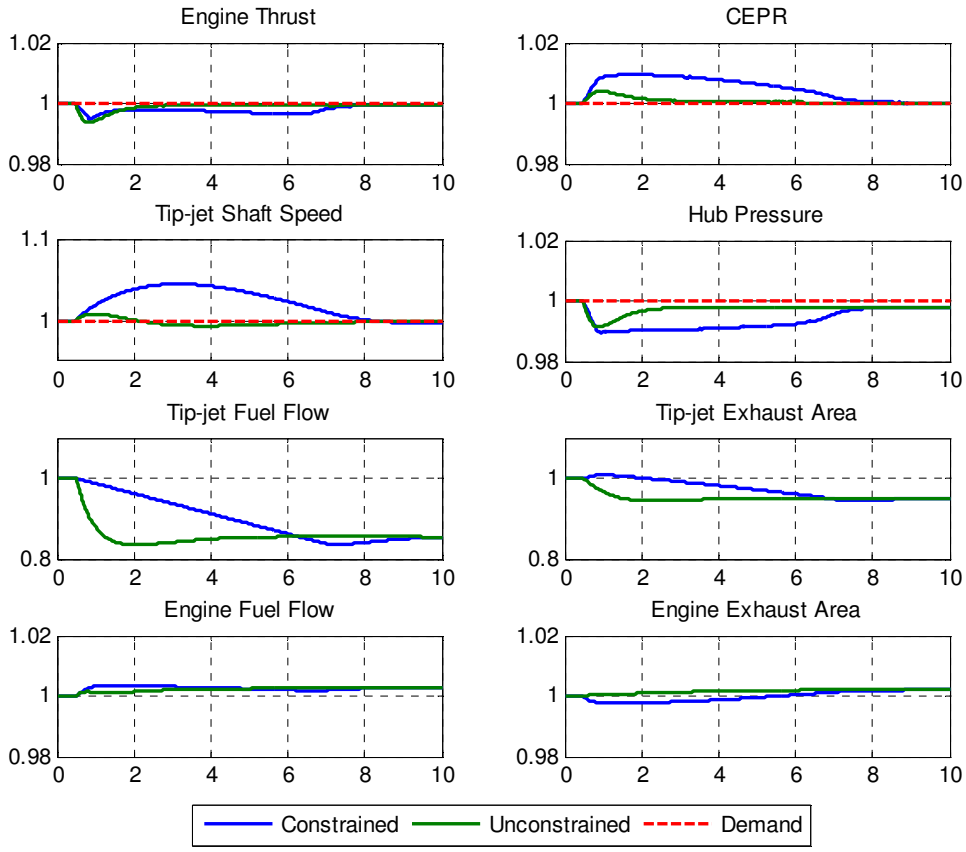

Figure 74: MPC Control Performance for Actuator Rate Limit Handling

7.5 **Centralized MPC Summary**

In this chapter, the centralized MPC controller for the tip-jet reaction drive system was developed. Before the controller was sized, components of the MPC architecture such as the state estimator were defined and various design variables were identified. A sensitivity study on the performance of the control to change in the design variables was performed. From this analysis, a final design selection was made. Integrating an optimizer in the design process could result in a final design variable setting that better matches the desired performance targets. Given this design, sensitivity to modeling error

and disturbance rejection study was performed. After this, different transient simulations were run that captured the response of the system varying throttle settings and changing rotor load. Lastly, simulations were run to understand the constraint handling capability of the controller. Given this baseline centralized MPC controller, the proposed feasible cooperative multiplexed MPC can be developed.

CHAPTER 8

DISTRIBUTED MODEL PREDICTIVE CONTROL

Now that the baseline PI controller and centralized MPC have been designed, the proposed distributed MPC can be developed. In the derivation of the proposed controller in CHAPTER 4 and CHAPTER 5, the proposed controller was referred to as a feasible cooperative multiplexed MPC. This added wording was necessary to differentiate it from both a feasible cooperative MPC and multiplexed MPC. In the next couple of chapters, the proposed control will only be compared with a centralized MPC therefore the extra feasible cooperative multiplexed notation is not needed. For simplicities sake, the proposed controller will be referred to as a distributed MPC from here on out.

8.1 **Distributed MPC Design Process**

As with the centralized MPC, the sizing and analyzing of the distributed MPC is a multi staged process. Figure 75 below shows a general step-by-step process that was followed to create the baseline distributed MPC controller. As with both the PI controller and centralized MPC design process, the distributed MPC design process can be broken up into three general categories: controller architecture components definition and modeling, controller design and sizing, and transient simulations. Within each of these general categories are multiple tasks.

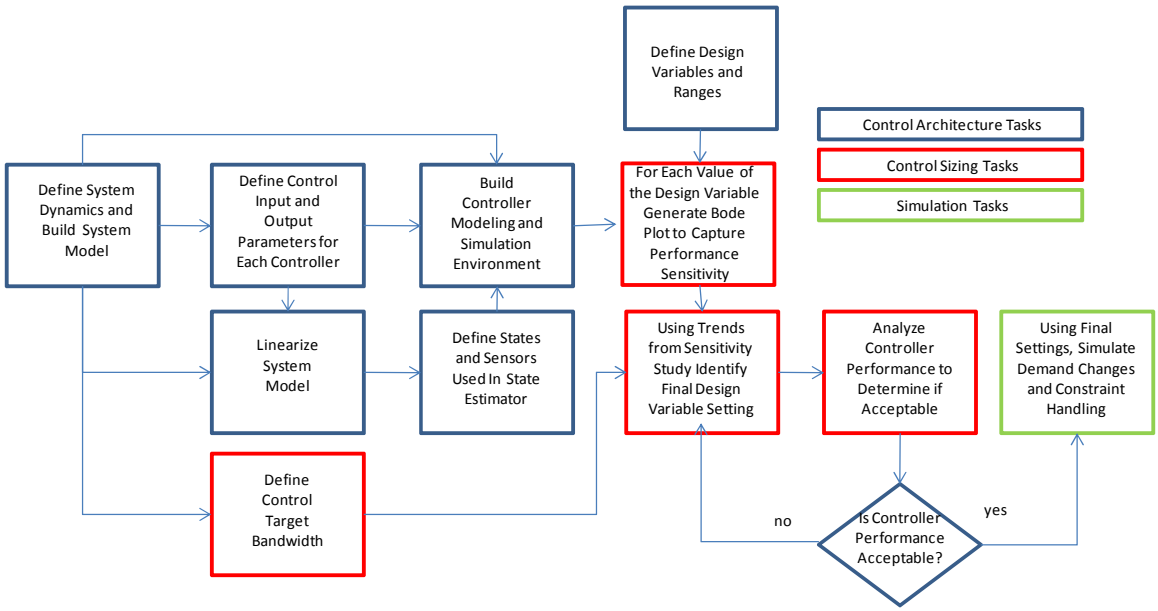

Figure 75: Distributed MPC Design Process

As with the centralized MPC, The first distributed MPC design category has six tasks. The first task is to define and quantify the dynamics of the system and build a non-linear system model of the tip-jet reaction drive system. This task is described earlier in detail in section 6.2.1. The second task is to linearize the tip-jet reaction drive system model. The linear tip-jet model is used both in the state estimator and in the distributed MPC algorithm. The third task is to define the appropriate control input and output parameters. As with the PI controller and centralized MPC, this task uses both the non-linear and linear models. The details of this task identical to that of the centralized MPC, therefore the centralized MPC discussion in section 7.2.1 can be used. The fourth task, which is described in section 7.2.2, is to define which states and sensors are to be used for the state estimator. After designing the state estimator, the next task is to define various elements of the distributed MPC algorithm including all the design variables and ranges. This discussion is covered in section 8.2.1. Similar to the centralized MPC, the distributed MPC needs to have the M&S environment defined before the controller can

be sized. The details of the distributed MPC M&S environment are discussed in section 8.2.2. Once these tasks are completed the controller can be sized.

Similar to the centralized MPC, the distributed MPC sizing process involves a sensitivity study to size the controller. The first task in the controller sizing category is to define the target bandwidth of the controller. For this dissertation, the target bandwidth was defined using the system dynamics outlined in the initial step in the control design process. The second task is to perform the sensitivity study which is done by varying each design variable across its range and then generating a Bode plot for each design variable setting. Unlike the centralized MPC, which used a linearized control model to generate the Bode plots, the Bode plots in created in the distributed MPC sensitivity study had to be generated by hand. This is required because the control environment is time variant and cannot be linearized. The Bode plots were generated by using varying frequency sinusoidal inputs and then taking Fourier transforms on the inputs and outputs to capture the phase and magnitude of the signals. Using the trends in the Bode plot, a final design variable selection can be made. The details of these two tasks are covered in section 8.3. Once the controller is sized, the last task is to analyze the performance of the controller. The details of this task are discussed in sections 8.3.1.7, 8.3.2, and 8.3.3. If the controller does not have acceptable performance properties, the sensitivity study results can be revisited to see if there is another suitable option.

Once a controller with acceptable performance qualities has been sized, various transient simulations can be analyzed. Using the M&S environment, simulations varying both throttle demand and rotor load changes can be run. The results of the simulations and discussion of the results is contained in section 8.4.1. The last simulation task to be performed is demonstration of PI controller constraint handling. The details of the constraint handling simulation are discussed in section 8.4.2. Once all these tasks are complete, the performance capabilities of a distributed MPC are understood and comparisons with other controllers can be made.

8.2 **Distributed Model Predictive Control Development**

Figure 76 below shows the distributed MPC architecture. There are some similarities and differences between this and the centralized MPC in Figure 39. Since the system dynamics, or available sensors, are not a function of the control architecture, both the state estimator and the onboard model, (G_{ob}), used on this architecture will be the same as that used for the centralized MPC. The main difference between the distributed and centralized MPC is the use of the three different MPC for the distributed control as opposed to a single MPC for the centralized control.

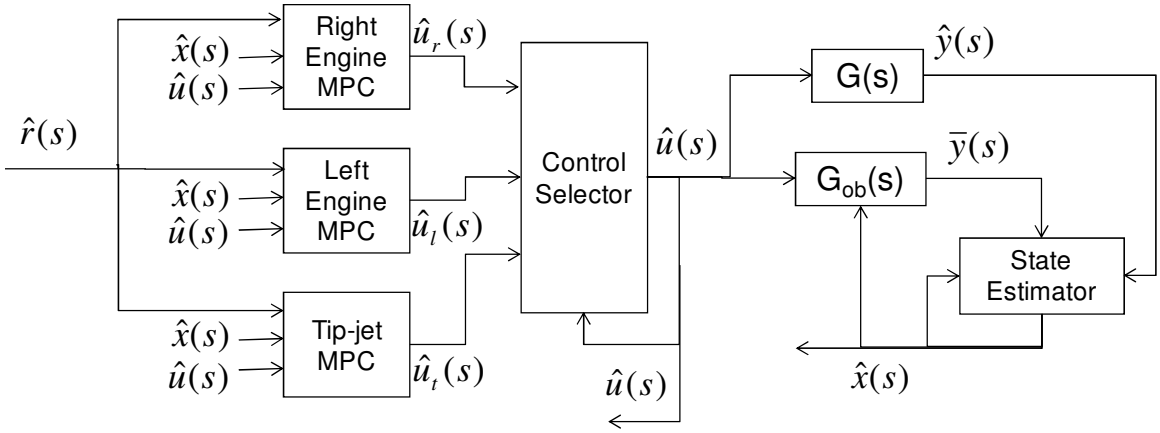

Figure 76: Distributed MPC Architecture

Although the state estimator and onboard model are identical to the centralized MPC, some aspects of the architecture need to be developed before the distributed MPC is designed. Since each distributed controller has a different set of control inputs and outputs, the MPC algorithm will be defined for each of the controllers. The control input/output selection analysis performed in the previous chapter will aid in the selection of control input/output parameters for each distributed controllers.

8.2.1 Distributed Model Predictive Control Algorithm

A generic distributed multiplexed MPC control algorithm for each MPC is defined below. The cost function is structured very similarly to the centralized MPC with both a reference tracking and control movement component. However, there are a few

significant differences. First each MPC only varies a single control input to minimize the cost function. This results in a significant reduction in the computational burden requirement to perform the cost function minimization. Secondly, the control input used in the optimization may be different at each time-step. And lastly, although the overall terms in the reference tracking portion of the cost function contain all control parameters, the weight attached to the reference tracking error of each parameter may vary.

$$V(x) = \sum_{n=1}^{N_d} \left[w_n \sum_{i=Hw}^{H_p} \left\| \hat{z}_n(k+i/k) - r_n(k+i) \right\|_{Q(i)}^2 \right] + \sum_{i=0}^{H_u-1} \left\| \Delta\hat{u}_{m,j}(k+i/k) \right\|_{R(i)}^2$$

Where

$\hat{z}_n$ are model calculated parameters for subsystem n

$\hat{r}_n$ are reference of target values for subsystem n

$\Delta\hat{u}_{m,j}$ are multiplexed actuator position movements for a particular subsystem MPC

$\hat{v}_j$ are measured disturbances to a particular subsystem MPC

H_p is the prediction horizon

H_u is the control horizon

H_w is the prediction horizon start point

Q and R are the cost function weighting matrices

w_n weighting factor for each subsystem cost function

Subject to these constraints

$$const < \hat{z}_j(t) < const$$
$$0 < \Delta\hat{u}_{m,j}(t) < const$$
$$const < \hat{u}_j(t) < const$$

Governed by state dynamics model

$$\hat{x}_j(t+1) = A\hat{x}_j(t) + B_j\hat{u}_j(t) + B_{d,j}\hat{v}_j(t)$$
$$\hat{z}_j(t+1) = C\hat{x}_j(t+1) + D_{d,j}\hat{v}_j(t)$$

Governed by control input equations

$$\hat{u}_{m,j}(t) = \hat{u}_{m,j}(t-1) + \Delta\hat{u}_{m,j}(t)$$

As with the centralized MPC, the onboard model is directly incorporated in the MPC algorithm through substitution of the state dynamics model and control input equations into the cost function. However, since only one control input is being optimized at a single instance, all the other non-optimized control inputs are handled as disturbances.

Given the above definition of the distributed MPC algorithm, various aspects of each distributed MPC algorithm can be quantified. The steady state model based design as well as the understanding of the system architecture can be used to define the variables in the $\hat{r}$ and $\hat{z}$ vectors used in each distributed MPC cost function. Similar to the centralized MPC, the weighting matrices and constraint definition, as well as other design variables specific to the distributed MPC architecture, are introduced and briefly discussed. As with the centralized MPC, the sensitivity of the control performance to these factors will be explored in more detail during the control sizing and simulation sections later in the chapter.

8.2.1.1 Control Input, Output, Reference, State Definition

The analysis in Section 6.1.1 was used to define which variables were used to populate the $\hat{r}$ and $\hat{z}$ vectors of the centralized MPC model. From the PI controller design, the tip-jet reaction drive system can be fairly naturally broken down into three subsystems: two engines and the tip-jet driven rotor. Each subsystem has a fuel flow and exhaust nozzle actuator. Using this subsystem breakdown coupled with the RGA analysis summarized in Table 14, each MPC controller can be broken down into 2x2 control input/primary control output couples. Since the other control parameters are also included in the MPC objective function, they are defined as secondary control parameters. Additionally all the non-optimized control inputs are defined in the

measured disturbance. The variables used in each MPC state vector are identical. At each control sampling instance, the state estimator will be used to update the system state, and feed the observed state into each MPC. Table 24, Table 25, and Table 26 summarize the control inputs, primary outputs, secondary outputs, measured disturbances, and states for the left engine, right engine, and tip-jet rotor MPC, respectively.

Table 24: Model Based Control Inputs, Outputs, References, States, and Disturbances for Left Engine MPC

Subsystem Primary Output and Reference, z_j and r_j	Subsystem Secondary Control Output and Reference, z_n and r_n	Control Inputs, $u_{m,j}$	State, x_j	Measured Disturbances, v_j
Left Engine Thrust Left Engine CEPR	Right Engine Thrust Right Engine CEPR Tip-Jet Rotor Shaft Speed Hub Pressure	Left Engine Fuel Flow Left Engine Exhaust Nozzle	Right LP Shaft Speed Right HP Shaft Speed Right Burner Metal Temperature Left LP Shaft Speed Left HP Shaft Speed Left Burner Metal Temperature Tip-jet Rotor Shaft Speed	Rotor Load Right Fan Flow Right Fan Efficiency Right Compressor Flow Right Compressor Efficiency Right HP turbine Flow Left Fan Flow Left Fan Efficiency Left Compressor Flow Left Compressor Efficiency Left HP turbine Flow Right Engine Fuel Flow Right Engine Exhaust Nozzle Tip-jet Fuel Flow Tip-jet Exhaust Nozzle

Table 25: Model Based Control Inputs, Outputs, References, States, and Disturbances for Right Engine MPC

Subsystem Primary Output and Reference, z_j and r_j	Subsystem Secondary Control Output and Reference, z_n and r_n	Control Inputs, $u_{m,j}$	State, x_j	Measured Disturbances, v_j
Right Engine Thrust Right Engine CEPR	Left Engine Thrust Left Engine CEPR Tip-Jet Rotor Shaft Speed Hub Pressure	Right Engine Fuel Flow Right Engine Exhaust Nozzle	Right LP Shaft Speed Right HP Shaft Speed Right Burner Metal Temperature Left LP Shaft Speed Left HP Shaft Speed Left Burner Metal Temperature Tip-jet Rotor Shaft Speed	Rotor Load Right Fan Flow Right Fan Efficiency Right Compressor Flow Right Compressor Efficiency Right HP turbine Flow Left Fan Flow Left Fan Efficiency Left Compressor Flow Left Compressor Efficiency Left HP turbine Flow Left Engine Fuel Flow Left Engine Exhaust Nozzle Tip-jet Fuel Flow Tip-jet Exhaust Nozzle

Table 26: Model Based Control Inputs, Outputs, References, States, and Disturbances for Tip-Jet MPC

Subsystem Primary Output and Reference, z_j and r_j	Subsystem Secondary Control Output and Reference, z_n and r_n	Control Inputs, $u_{m,j}$	State, x_j	Measured Disturbances, v_j
Tip-Jet Rotor Shaft Speed Hub Pressure	Right Engine Thrust Right Engine CEPR Left Engine Thrust Left Engine CEPR	Tip-jet Fuel Flow Tip-jet Exhaust Nozzle	Right LP Shaft Speed Right HP Shaft Speed Right Burner Metal Temperature Left LP Shaft Speed Left HP Shaft Speed Left Burner Metal Temperature Tip-jet Rotor Shaft Speed	Rotor Load Right Fan Flow Right Fan Efficiency Right Compressor Flow Right Compressor Efficiency Right HP turbine Flow Left Fan Flow Left Fan Efficiency Left Compressor Flow Left Compressor Efficiency Left HP turbine Flow Right Engine Fuel Flow Right Engine Exhaust Nozzle Left Engine Fuel Flow Left Engine Exhaust Nozzle

8.2.1.2 Prediction Horizon and Prediction Time-step

Since the dynamics of the system do not change with the new distributed control architecture, both the prediction horizon and model time-step defined in Chapter 7 can be used. The number of control horizon steps should be of similar order to the prediction horizon. However, since each control input is optimized every other time-step, the number of control inputs being optimized would be at most half the size of the prediction horizon. In the centralized MPC sizing, it was seen that a control horizon much shorter than the prediction horizon would provide virtually identical control performance to a one the same size as the prediction horizon. Understanding that fact, a similar sensitivity

study of the effect of control horizon on the distributed MPC performance will be performed.

8.2.1.3 Q,R, and w_o Weighting Factors

The Q and R weighting factors in the cost function are variables which weight the importance of matching the reference target versus minimizing control input movements. For the MPC algorithm to be stable and solvable, the values of the weighting factors need to be greater than or equal to zero for all points along the prediction and control horizon.

As mentioned previously, Q is a weighting factor applied to the reference matching portion of the MPC cost function. Besides the greater than or equal to zero constraint, Q can hold any value. For this set of experiments Q was assumed to be time variant. However to limit the number of potential variables the same weighting factor is applied to each control parameter, j. Q can also be used to capture to effect of H_w by setting all values of Q(i) equal to zero for all i that are less than H_w.

R is a weighting factor for the movements of the control input portion of the MPC cost function. Similar to Q, the only fixed criteria for R is that is be greater than or equal to zero for points along the control horizon. R acts as a damper on the response where the larger the value of R, the slower the response.

Of more importance than the absolute values of R and Q is the relative value of each when compared with each other. As the value of R becomes significantly larger than Q, the control moves much slower to the extent that the control inputs would not move at all if R was significantly larger than Q.

Lastly integrating all the subsystem cost functions into each subsystem MPC controller may lead to interactions. A weighting factor, w_o, on the other subsystem objective is introduced to potentially minimize any encountered interactions. In the MPC sizing section in section 8.3.1, various combinations of Q R, w_o, as well as values of H_w will be explored to obtain the desired MPC performance.

8.2.1.4 Fuel Syncing Strategy

Another potential parameter that may affect the performance of the control is the multiplexing order. For each subsystem there are 2 different potential ways to multiplex the control (assuming the multiplexing is just alternating the variables each time-step). For the tip-jet reaction drive system, the total numbers of combinations is 8. Numerous other strategies such as variable frequency scheduling could also be explored.

However, as will be discussed later the CPU requirements to explore the design space for the distributed MPC architecture are quite significant making the exploration of all 8 combinations infeasible. Syncing up the engine order of multiplexing will ensure there are no disturbances added to the system because the engine subsystem controllers are out of sync, therefore making any combination with the engine controllers out of sync irrelevant. This leaves just two schemes that could be explored: tip-jet fuel synced with engine fuel and tip-jet fuel out of sync with engine fuel flow. These two combinations may have a different effect on the control performance and each will be explored in the subsequent control sizing section.

8.2.1.5 Handling of Non-Optimized Inputs

The non-optimized terms are integrated into the distributed MPC algorithm as a known disturbance. How they are handled may have an effect on how well the algorithm performs. The first option is to keep the values constant throughout the prediction horizon. Although simple this option prevents the system from ever operating at the optimized path from the previous time-step. Another option would be to use the pre determined optimized path from the last optimization of that variable. Neither of these methods appears to affect the computational burden of the MPC algorithm and both will be explored in the control sizing section.

8.2.1.6 Number of Iterations

In the feasible cooperative strategy the cost function optimization is performed every iteration and the information from the other control inputs is updated after each iteration. If there are significant interactions, multiple iterations may be required to obtain optimal performance, at the cost of increased computational burden. To minimize the number of iterations, the sensitivity of the control performance to the number of iterations should be explored. Knowledge of this relationship would prevent the MPC designer from setting the number of iterations in the algorithm too high, and in turn adversely affecting the computational burden.

8.2.1.7 Tip-jet Reaction Drive Constraints

The constraints present on the distributed controller are identical to those encountered for the centralized MPC. The main difference is how the controllers integrate the constraints. Because of potential numerical issues only the constraints affected by the control input being optimized are included in the specific controller. This will be further discussed in the constraint handling section. Additionally, since only one control input is being optimized at a time, only a single actuator position and rate constraint are needed.

8.2.2 Distributed MPC Modeling and Simulation Environment

Using the architecture shown in Figure 76 as a guideline, the distributed MPC modeling and simulation (M&S) environment was developed using Simulink. For this controller architecture, there are five major components that need to be modeled: the tip-jet reaction drive system, G(s), which transposes control inputs to measured output, the onboard model, $G_{ob}(s)$, which is used to estimate unmeasurable system parameters, the state estimator which is used to both estimate the states of the system and ensure the onboard model matches the measured parameters of the system, each distributed MPC

which determines the control inputs for each subsystem required to minimize a locally defined objective function, and a control input selector which is used to define which control inputs should be sent to the tip-jet reaction drive system. As was done for the PI controller, the tip-jet reaction drive system is represented by the non-linear NPSS defined earlier in section 6.2.1.3. The NPSS model is executed from within Simulink using a custom dynamic link library (DLL) developed by NASA. All the inputs and outputs into this model are normalized around the design point to ensure good matrix properties for the controller design.

The onboard model is represented by the linearized version of the normalized inputs and outputs of the NPSS model. Normally, multiple linear models would be created representing multiple operating points and power settings with these models the being curve fit together. However since the analysis is done around a single operating point, only a single linear model was used for the onboard model.

The state estimator uses the extended Kalman filter defined in section 7.2.2.4 to estimate the state of the tip-jet reaction drive system. As was discussed in the state estimator design section performed in section 7.2.2.3, the state estimator uses the linearized onboard model, but this model will be augmented by data match scalar states. This ensures that both the onboard model output matches the actual output of the tip-jet reaction drive system and the controller has offset free reference tracking. To simulate the state estimator, a custom Simulink S-Function, whose source code is located in Appendix D, was created that contained all the extended Kalman filter logic.

Each distributed MPC controller also uses the linearized onboard model augmented by data match scalars. As opposed to the centralized MPC, which was simulated using a Simulink MPC element, each distributed MPC is simulated using a custom Simulink S-Function using the Matlab function ‘mpcmove’, which simulates a single time-step of an MPC algorithm. In addition to the ‘mpcmove’ function, there is logic in each distributed MPC S-Function to determine which control inputs are to be

optimized and defined as measured disturbances. The source code for the distributed MPC S-function is located in Appendix E.

The control selector which defines what the appropriate control inputs are for each time-step has three functions. The first is to select only the first time-step value from the results of the distributed MPC. The second is to store the selection results from the previous time-step which represent the control input not being optimized at the current time-step. The last step is to combine the values of the previous two steps into a single array. This array represents the control inputs for the tip-jet reaction drive system for the current time-step. The execution of the Simulink based distributed MPC M&S environment is done from a Matlab command prompt. Now that the all the components of the distributed MPC architecture have been defined and the M&S environment has been developed, the distributed MPC can be sized.

8.3 **Distributed Model Predictive Control Design**

The metrics for the decentralized MPC controller performance are virtually identical to that of the PI controller and centralized MPC. For engine thrust and CEPR the target bandwidth is approximately 3 rad/s. For the tip-jet rotor speed, the target bandwidth is around 0.6 rad/s. The hub pressure bandwidth should be similar to the engine bandwidth of 3 rad/s. The interactions should be minimal across all frequencies. For stability concerns, gain margin and phase margin should be greater than 6 dB and 45^{o}, respectively.

8.3.1 Control Sensitivity to Design Variables

As mentioned in the previous section, the decentralized MPC design variables are as follows: 1.) the prediction horizon start point for each variable in the cost function, 2.) the cost function weighting factors Q and R, 3.) the control horizon, 4.) other subsystem cost function weighting, 5.) fuel flow syncing strategy, 6.) handling of non optimized

terms, and 7.) number of iterations. The sensitivity of the control performance metrics to each of these variables will be discussed and analyzed next. The final design of the centralized MPC was used as a starting point for the sensitivity study. From this point, each of the design variables can be changed and the results analyzed with the goal of finding a design that meets the control performance metrics. For reference, the baseline point values for each design variables are: H_{we} = 15, H_{wr}= 80, R = 2.5, Q = 0.5, w_o = 0.1, H_p =100, H_u = 100; In addition, non-optimized terms are assumed constant over time, and a single iteration.

In the subsequent Bode analysis, each column in the Bode plot represents the change in demand, whereas each row represents the response of the parameter to a change in the aforementioned demand. Similar to the centralized MPC, shorthand notation as shown in Table 27 will be used to capture the response of different control parameters to changes in demand. And was done in the previous chapters, the green circles will highlight the bandwidth sensitivity while the red squares highlight the largest interactions.

Table 27: Distributed MPC Bode Plot Descriptions

Bode Plot Column and Row Description	Column Description	Row Description
Fg Right	Right engine thrust demand change	Right engine thrust response to a demand change
CEPR Right	Right engine CEPR demand change	Right engine CEPR response to a demand change
Fg Left	Left engine thrust demand change	Left engine thrust response to a demand change
CEPR Left	Left engine CEPR demand change	Left engine CEPR response to a demand change
Tip-jet Shaft	Tip-jet shaft speed demand change	Tip-jet shaft speed response to a demand change
Hub Pressure	Hub pressure demand change	Hub pressure response to a demand change

Generating the Bode plots for the decentralized MPC was much tougher than for the centralized MPC. The centralized MPC was time invariant and thus the control

system could be linearized around an operating point, making the sensitivity study not very CPU intensive. Since the decentralized MPC is time variant, the control could not be linearized. Therefore the Bode plots for each design point had to be generated by hand which was a time consuming two step process. The first step was to generate sinusoidal inputs at different frequencies for each of the demand inputs while also capturing the control output response to each input. The second step was to take a Fourier transform of both the sinusoidal inputs and outputs and capture the magnitude and phase of each. This process increased the CPU requirements for a single design point analysis from about 5 seconds to around 12 hours, which limited the size of the design space that could be explored effectively. In the subsequent analysis, a much smaller design space relative to the centralized MPC will be explored. Unfortunately, in selecting a final design variable setting, some extrapolation of design variable trends was involved.

The stability analysis of the distributed MPC was also limited because the Bode plots were generated by hand. At higher frequencies, the calculation of phase does not appear very reliable making the estimation of gain margin, and at times phase margin, almost impossible. Therefore in this analysis, the sensitivity of gain margin to changes in the different design variables will not be explored.

8.3.1.1 Prediction Horizon Start Point

Figure 77 and Figure 78 below show Bode plot sensitivity to changes in the engine prediction horizon start point, H_{we}. The engine prediction horizon start point ranged from a value of 30 to 1. For all the control except tip-jet rotor speed, as H_{we} increases, the bandwidth decreases, however hub pressure appears changes much more significantly relative to the other variables. The engine bandwidth ranged from 0.7 to 1.1 rad/s. The CEPR bandwidth ranged from 1.6 to 2.1 rad/s. While the hub pressure ranged

from very small to 3.8 rad/s. Using just prediction horizon start point, only the hub pressure can achieve the target bandwidth.

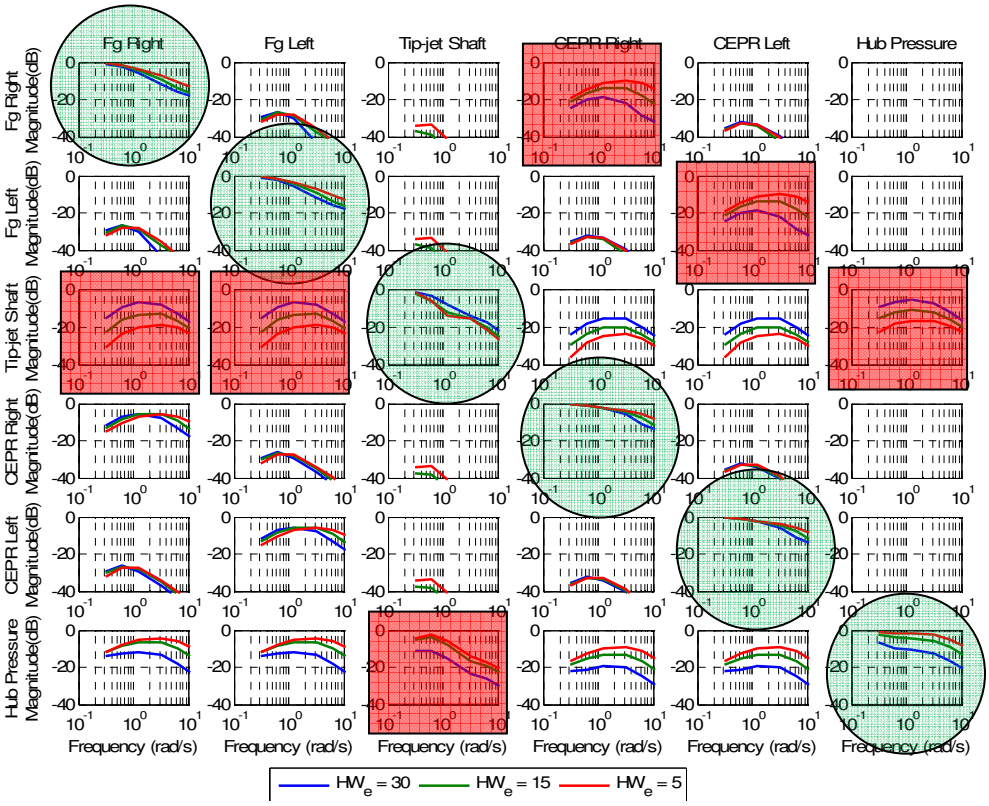

Figure 77: Bode Plot Magnitude Sensitivity to Changes in Engine Prediction Horizon Start Point

As shown in Figure 77, the hub pressure interactions with a tip-jet shaft speed demand change become significant when prediction horizon start point is small. Overall decreasing H_{we} causes hub pressure interactions with change in demand of all the other control parameters. Tip-shaft speed interactions with changes in hub pressure and engine thrust demand become quite significant when the prediction horizon start point is high. Another interaction that is fairly sensitive to engine prediction horizon start point is thrust with changes to CEPR demand. The phase margin appeared to be around 140 for all the runs and was not significantly affected by changes in the prediction horizon start point.

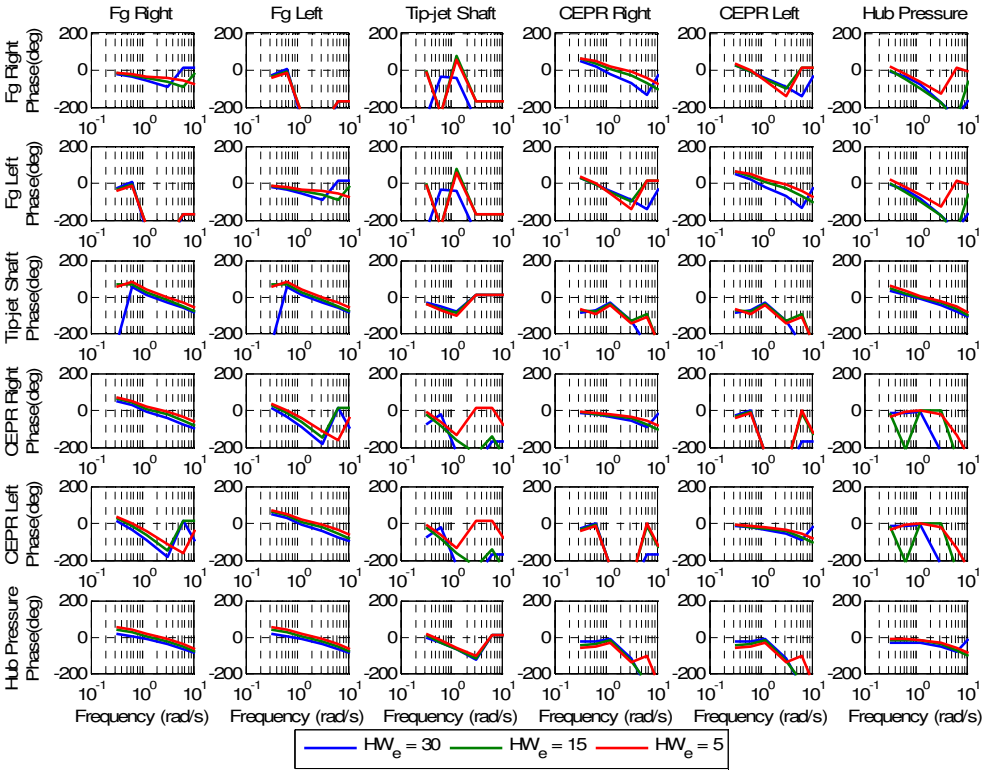

Figure 78: Bode Plot Phase Sensitivity to Changes in Engine Prediction Horizon Start Point

Figure 79 and Figure 80 below show Bode plot sensitivity to changes in the rotor prediction horizon start point, H_{wr}. Using the engine prediction horizon start point analysis as a reference point, the range for H_{wr} was chosen to be around ¾ of the tip-jet rotor speed time constant of 100 time-steps. As expected the only control parameter bandwidth to be significantly affected by H_{wr}. Similar to H_{we} , as H_{wr} increases, the bandwidth decreases. The rotor bandwidth ranged from 0.2 to 0.5 rad/s. Using just prediction horizon start point, the rotor speed cannot achieve the target bandwidth but can get fairly close.

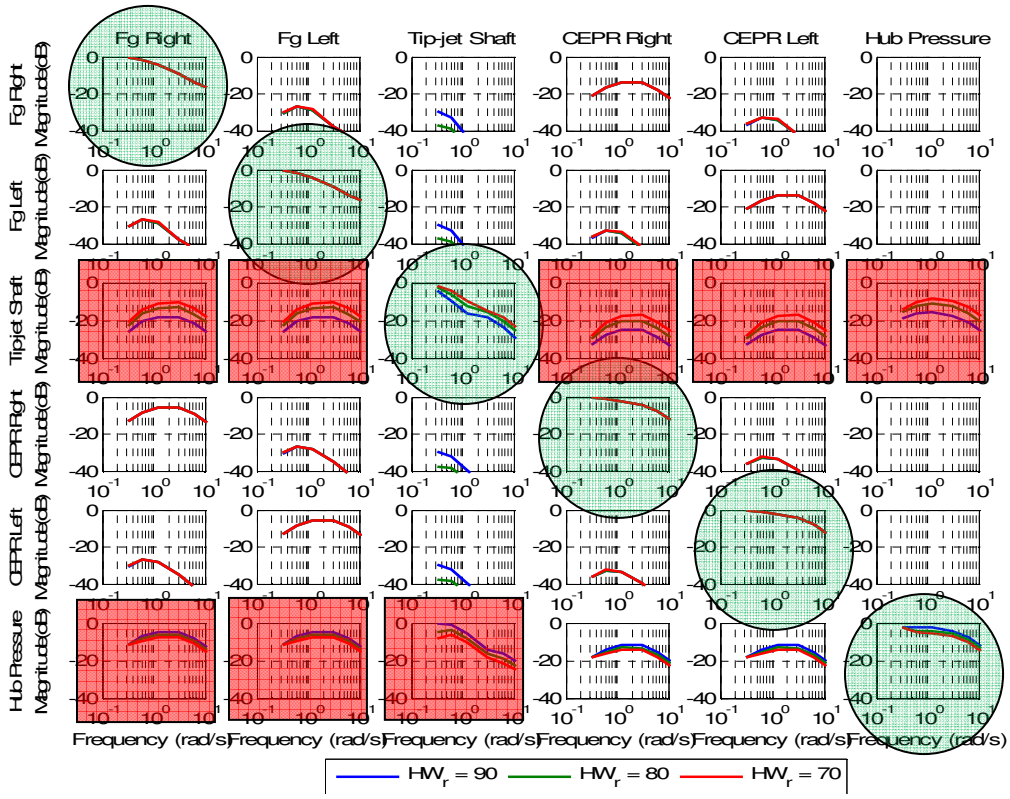

Figure 79: Bode Plot Magnitude Sensitivity to Changes in Rotor Prediction Horizon Start Point

As shown in Figure 79 at low values of H_{wr}, tip-jet shaft speed interactions with changes in demand of all the other control parameters become more significant. Hub pressure interactions with both thrust and tip-jet speed demand changes are generally fairly high. None of the other interactions were significantly affected by changes in the rotor prediction horizon start point. H_{wr} did not have much of an effect on the phase margin.

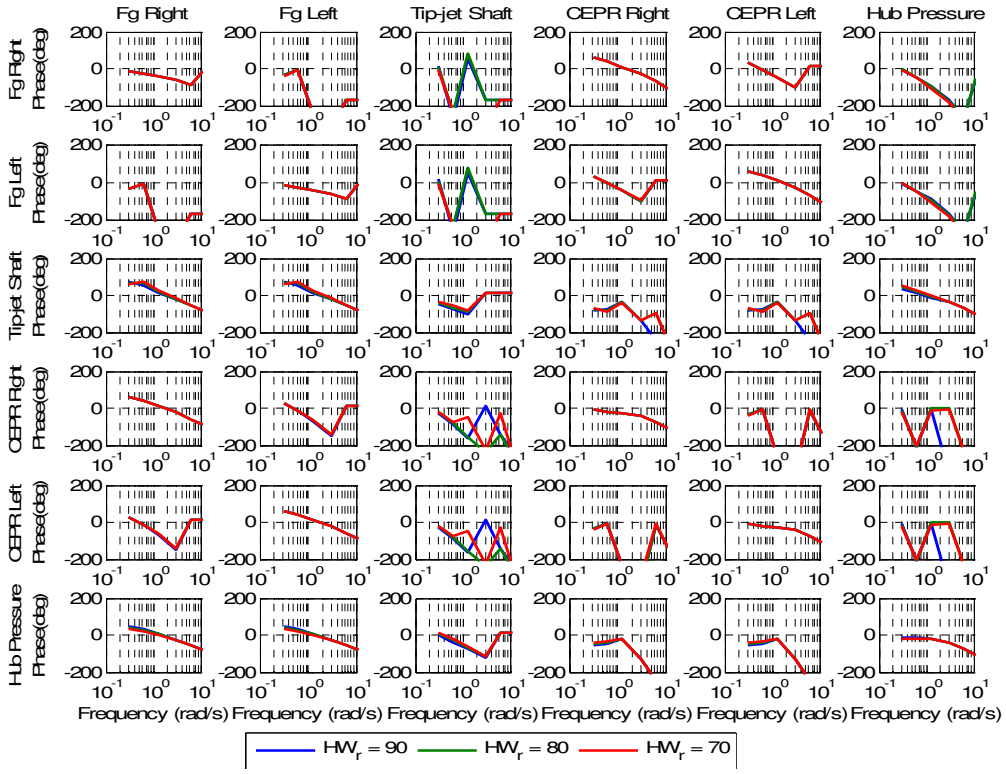

Figure 80: Bode Plot Phase Sensitivity to Changes in Rotor Prediction Horizon Start Point

8.3.1.2 Cost Function Weighting Factors

Figure 81 and Figure 82 below show Bode plot sensitivity to changes in the MPC cost function control movement weighting factor, R. The range of R was from 10 times as large a Q (R=2.5) to twice as large (R=1.0). As R increases, the bandwidth of the all parameters decreases. Engine thrust bandwidth ranged from 0.6 to 1.8 rad/s. Tip-jet rotor speed bandwidth varied from 0.2 to 0.7 rad/s. CEPR bandwidth ranged 0.9 to 6.34 rad/s. While hub pressure bandwidth ranged from 0.4 to 0.6 rad/s. Only tip-jet shaft speed and CEPR fall within the bandwidth targets. Both thrust and hub pressure are slower than the targets.

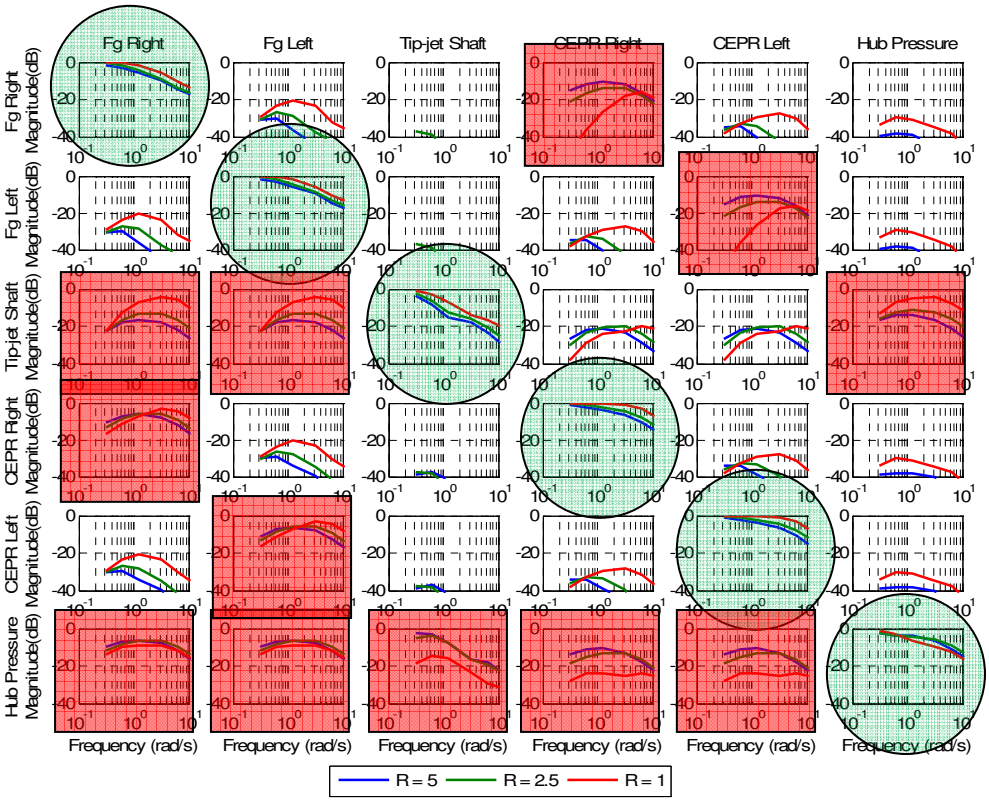

Figure 81: Bode Plot Magnitude Sensitivity to Changes in Control Movement Weighting Factor

Figure 81 also captures the sensitivity of the interactions to changes in R. As R increases, the thrust interactions with CEPR demand changes increases. Low values of R cause significant increases in tip-jet shaft speed interactions with change in thrust and hub pressure demand. Independent of changes in R there are fairly high interactions of CEPR with changes in thrust demand. Lower values of R reduce the amount of interactions of hub pressure with changes in the demand of the other parameters.

For all parameters except tip-jet shaft speed phase margin decreased a fair amount as R decreased. Thrust went from 150 to 129. Tip-jet shaft speed was approximately 135. CEPR decreased from 153 to 120. While hub pressure decreased from 164 to 146.

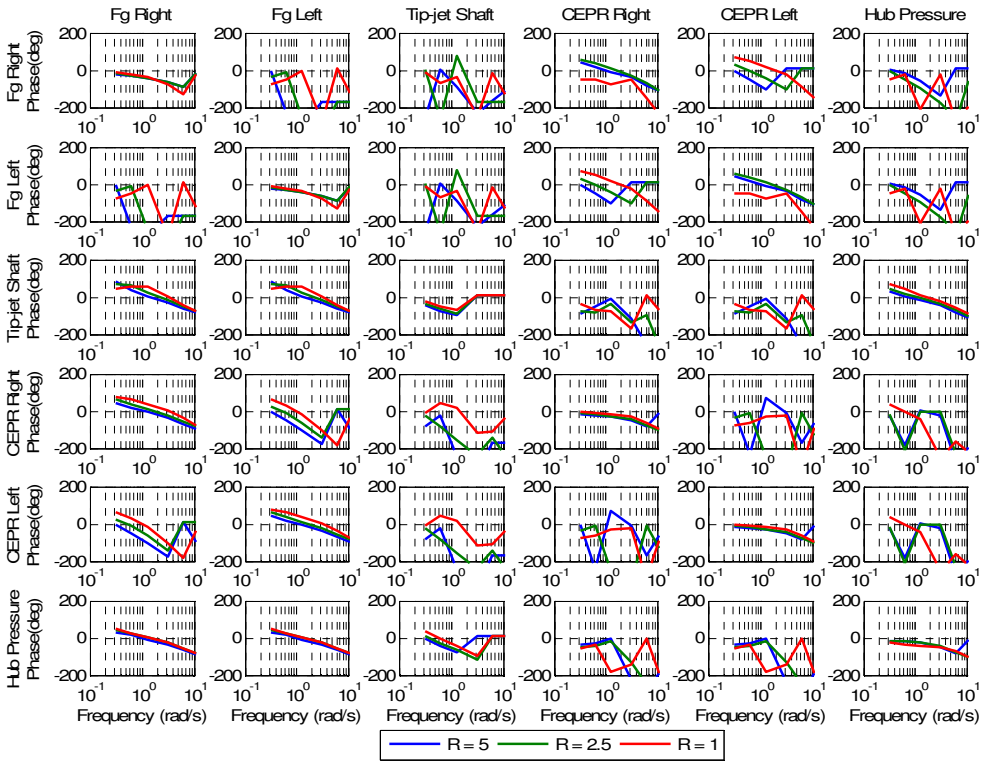

Figure 82: Bode Plot Phase Sensitivity to Changes in Control Movement Weighting Factor

Figure 83 and Figure 84 below show Bode plot sensitivity to changes in the MPC cost function reference tracking weighting factor, Q. The range of Q was from 2.5 times as large as R (Q=1) to 10 times as small (Q=0.1). As expected Q has the opposite effect as R. As Q increases, the bandwidth of the all parameters except hub pressure increases. Engine thrust bandwidth ranged from 1.6 to 0.6 rad/s. Tip-jet rotor speed bandwidth varied from 0.5 to 0.2 rad/s. CEPR bandwidth ranged from 4.6 to 0.9 rad/s. While hub pressure bandwidth ranged from 0.46 to 0.44 rad/s. Besides CPR, the bandwidth of all the control parameters falls below the target bandwidth.

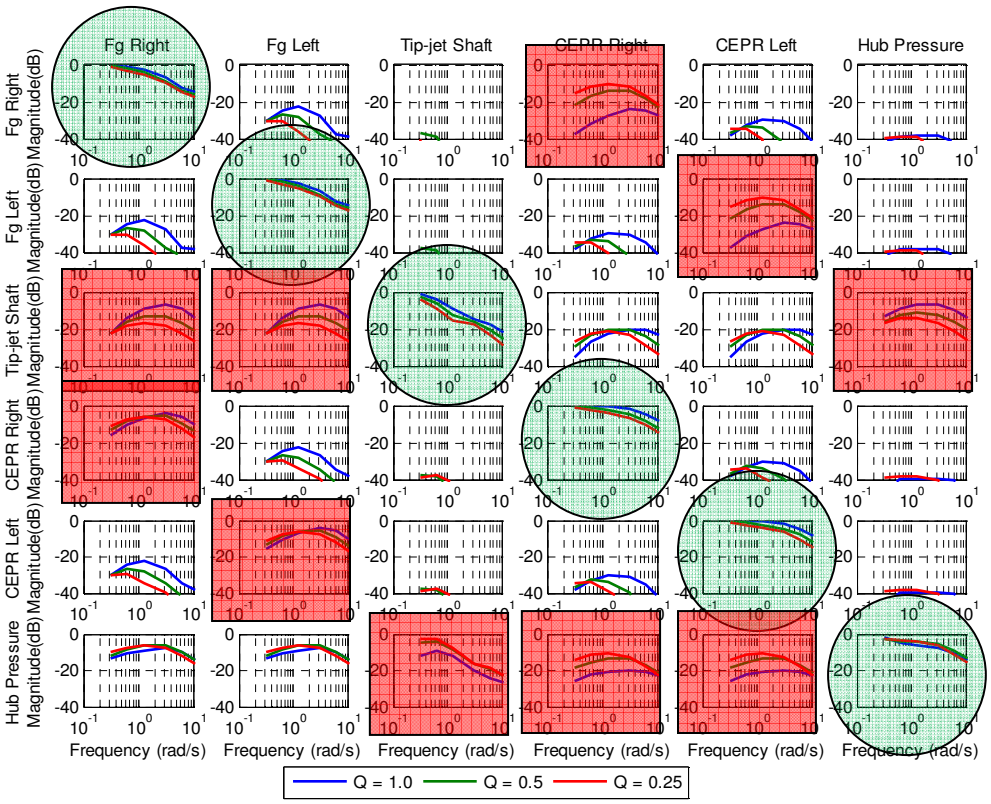

Figure 83: Bode Plot Magnitude Sensitivity to Changes in Reference Tracking Weighting Factor

Figure 83 also captures the sensitivity of the interactions to changes in Q. As Q decreases, the thrust interactions with CEPR demand changes increases. High values of Q cause significant increases in tip-jet interactions with change in thrust and hub pressure demand. Independent of changes in Q there are fairly high interactions of CEPR with changes in thrust demand. Higher values of R reduce the amount of interactions of hub pressure with changes in demand of the other parameters.

For all parameters except tip-jet shaft speed phase margin increased a fair amount as R decreased. Thrust went from 133 to 151. Tip-jet shaft speed was approximately 135. CEPR increased from 131 to 153. While hub pressure decreased from 154 to 164.

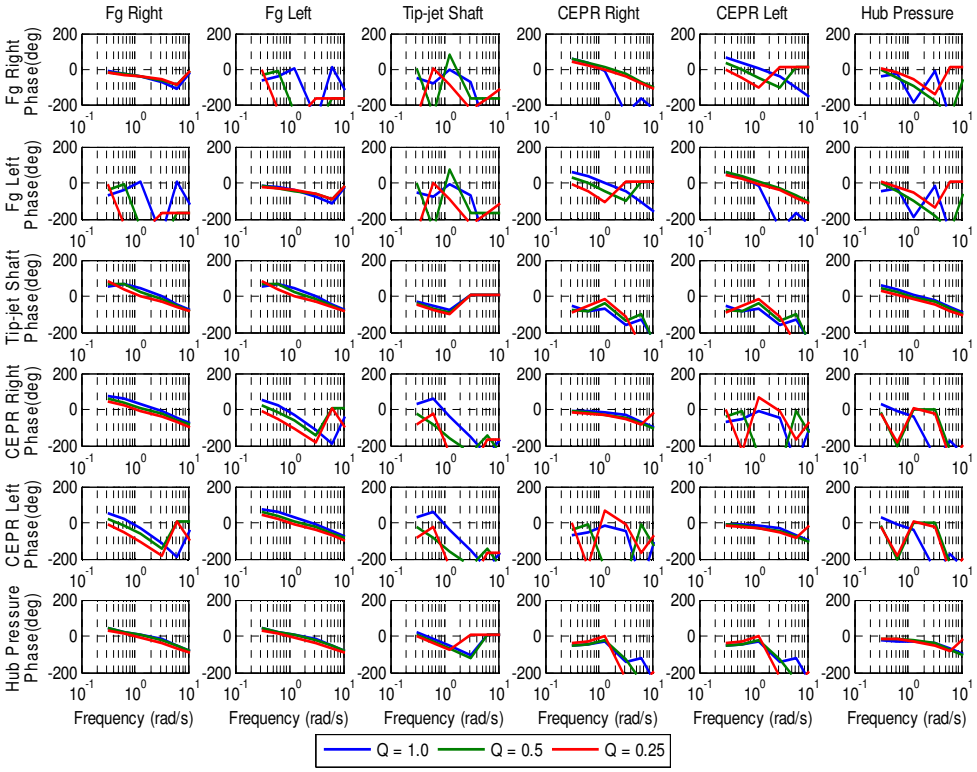

Figure 84: Bode Plot Phase Sensitivity to Changes in Reference Tracking Weighting Factor

Figure 85 Figure 86 below show Bode plot sensitivity to changes in the weighting factor of other subsystem objectives. Besides hub pressure, the bandwidth of all the control parameters was unaffected by w_o. At high values of w_o the hub pressure bandwidth becomes significantly lower than target. In general, the higher the value in the weighting factor, the larger the interactions are. Therefore, small values of w_o are required to maintain low levels of interactions. W_o had no effect on phase margin.

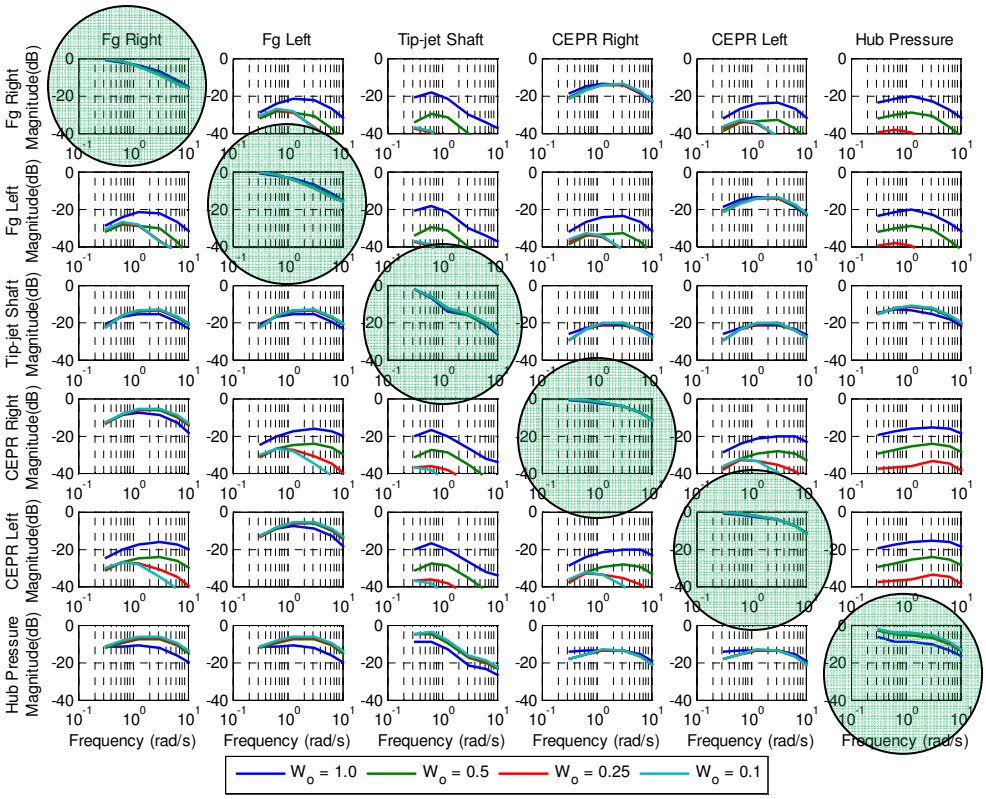

Figure 85: Bode Plot Magnitude Sensitivity to Changes in Other Subsystem Weighting Factor

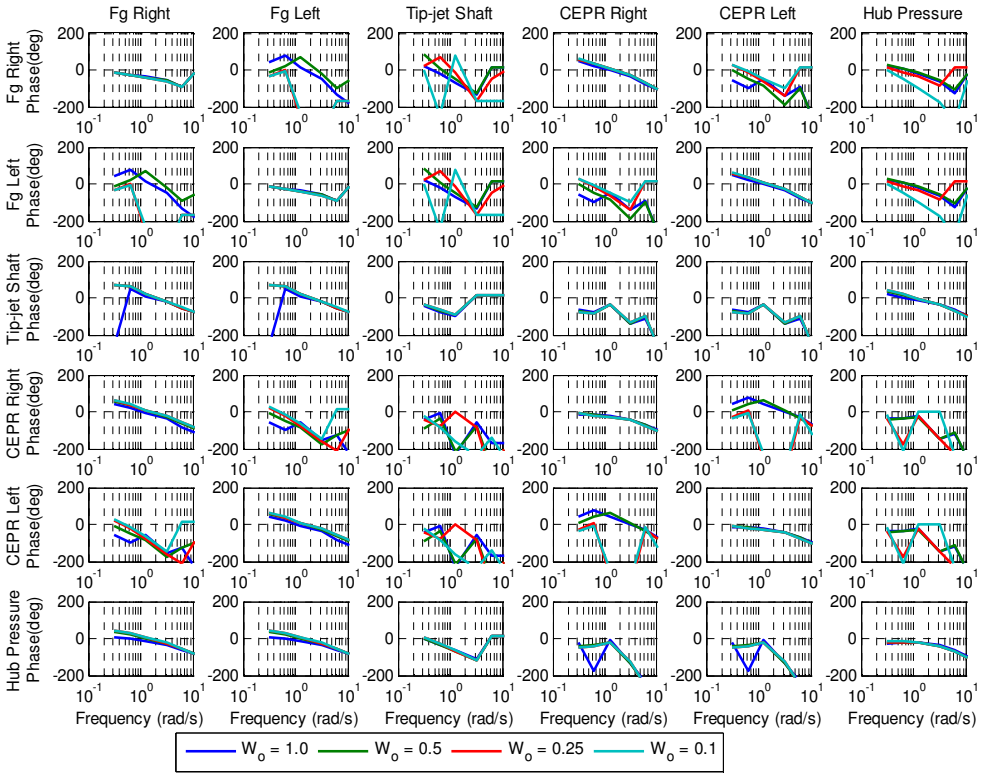

Figure 86: Bode Plot Phase Sensitivity to Changes in Other Subsystem Weighting Factor

8.3.1.3 Control Horizon

Figure 87 and Figure 88 below show Bode plot sensitivity to changes in the control horizon. The control horizon ranged from equal the prediction horizon (H_u=100) to just a handful of time-steps (H_u=20). Because of the alternating of control inputs, the actual number of control values being optimized is half of the value of the control horizon. The control horizon did not have a significant effect on the bandwidth or phase margin for all the control parameters. However, it did have a fairly significant effect on the bandwidth of CEPR and hub pressure. For both of these parameters, as control horizon decreased, bandwidth increased. Therefore since computational burden is proportional to the control horizon, the smallest value of the control horizon is chosen. This is a very positive observation since the computational burden of the MPC is proportional to the control horizon.

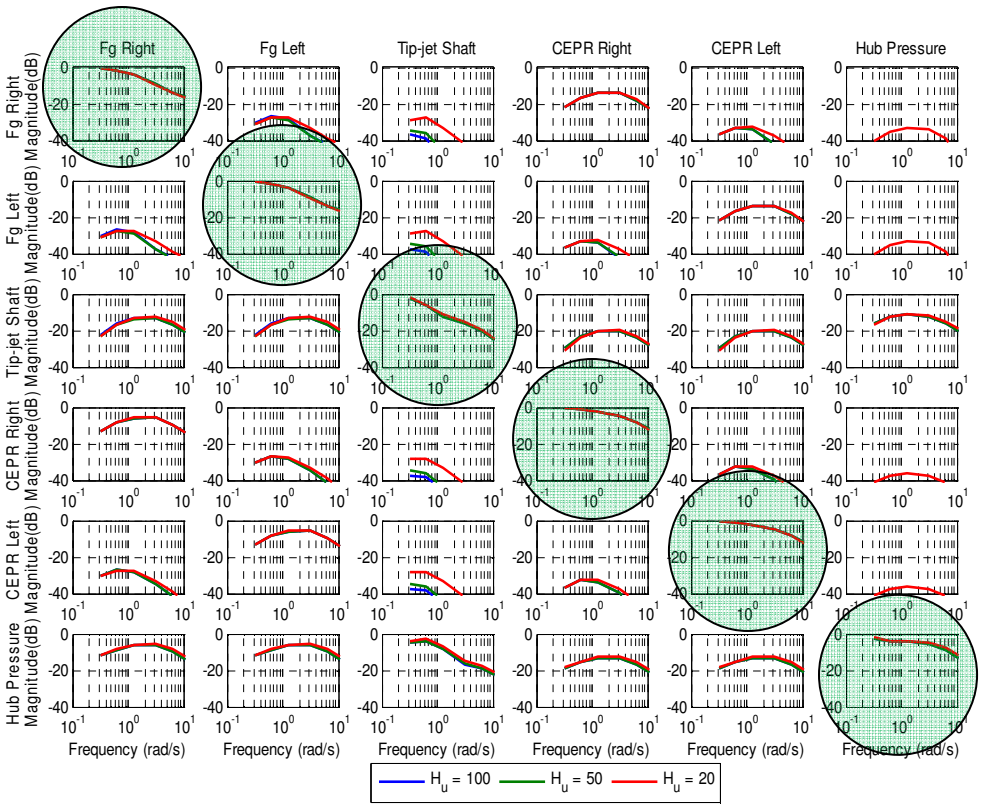

Figure 87: Bode Plot Magnitude Sensitivity to Changes in Control Horizon

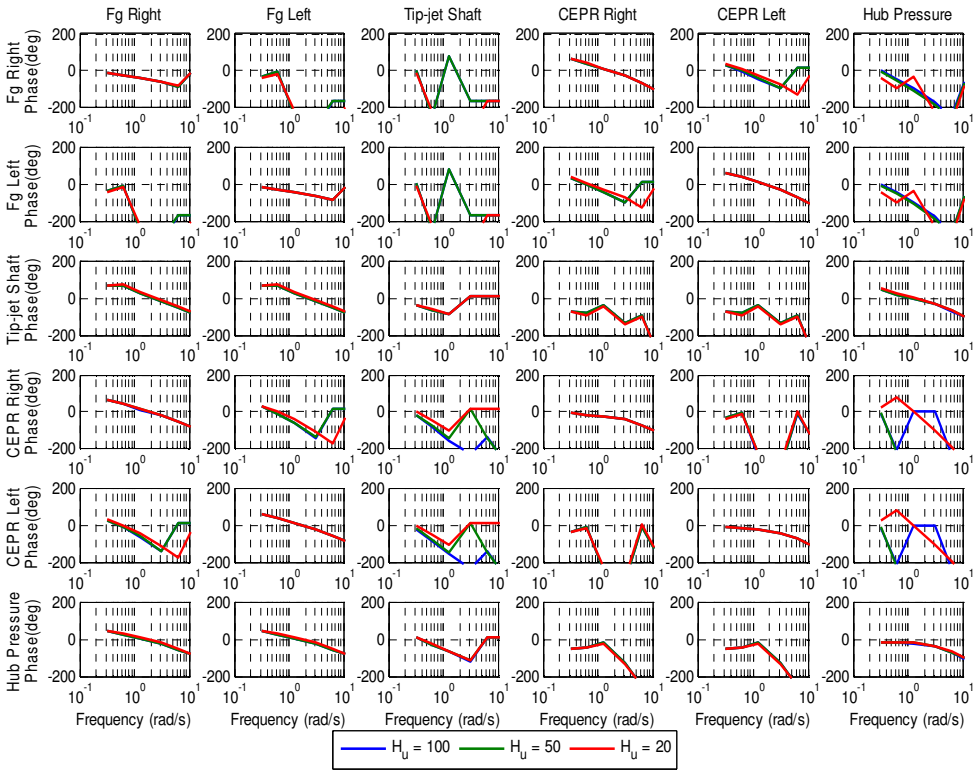

Figure 88: Bode Plot Phase Sensitivity to Changes in Control Horizon

8.3.1.4 Fuel Syncing

Figure 89 and Figure 90 below show Bode plot sensitivity to changes in the fuel syncing strategy. Only the tip-jet shaft speed and hub pressure bandwidth was affected by the fuel syncing strategy with tip-jet shaft speed bandwidth decreasing and hub pressure bandwidth increasing when all the fuel controls are in sync. Alternating the fuels causes the tip-jet reaction drive interactions with changes in the demand of all the other control parameter to increase. However, when the fuels are alternated, the hub pressure interaction with tip-jet shaft speed demand change becomes smaller.

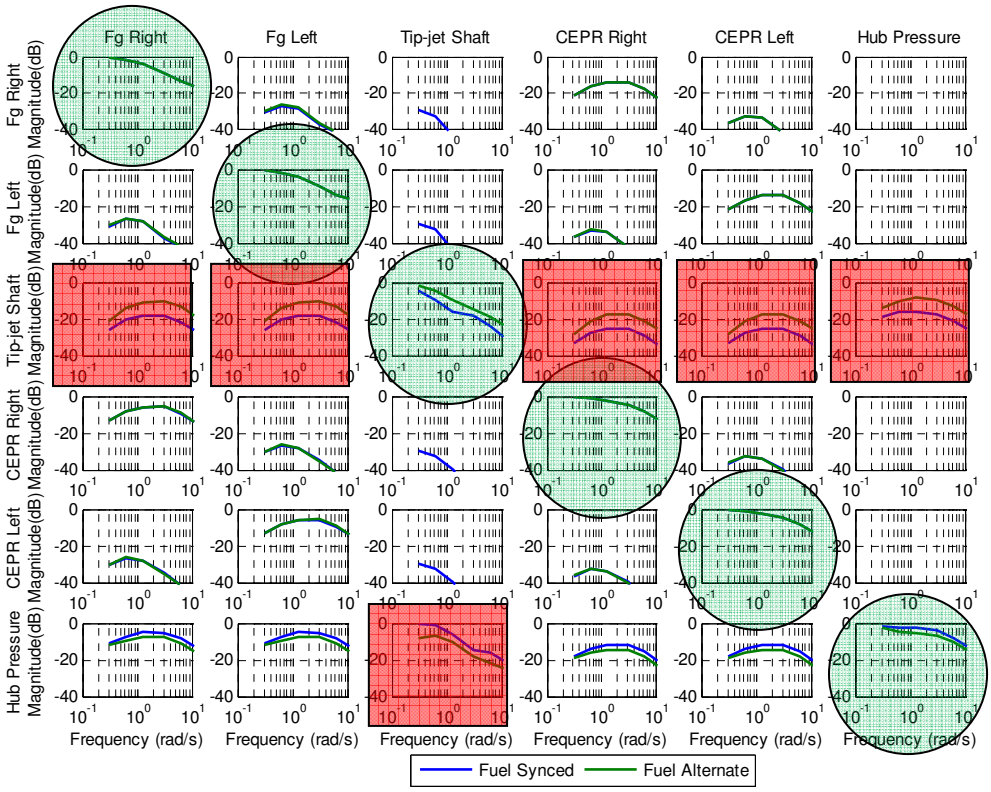

Figure 89: Bode Plot Magnitude Sensitivity to Fuel Syncing Strategy

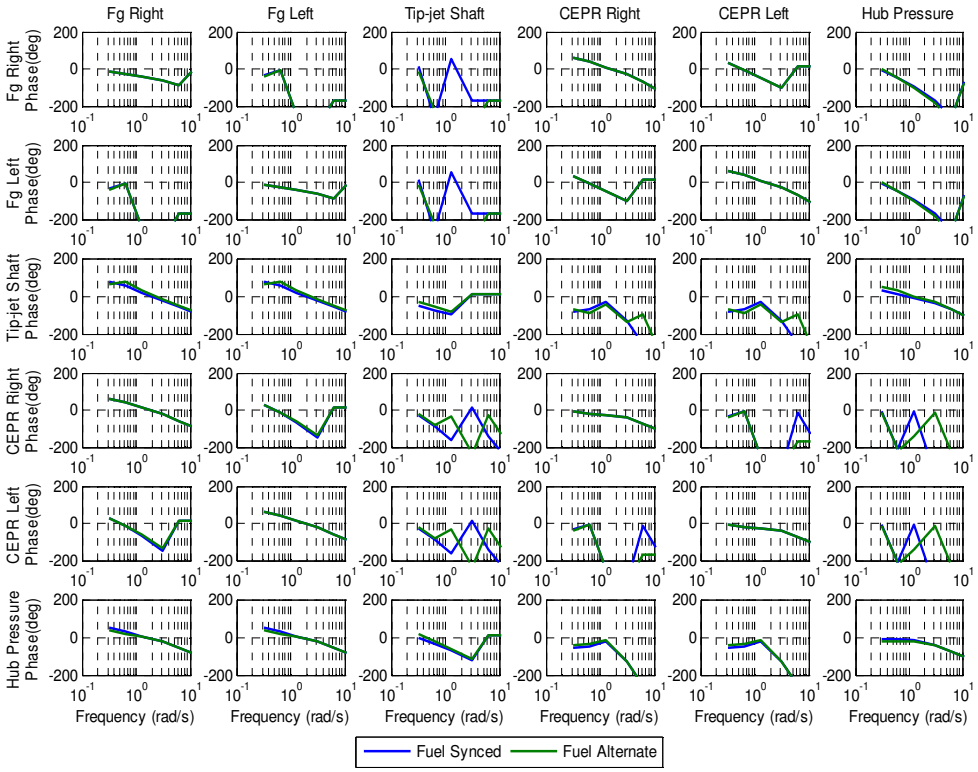

Figure 90: Bode Plot Phase Sensitivity to Fuel Syncing Strategy

8.3.1.5 Handling of Non-optimized Control Inputs

Figure 91 and Figure 92 below show Bode plot sensitivity to changes the handling of non-optimized terms. The bandwidth and phase margin of all the control parameters is not significantly changed. However, when the full optimized path from the last input is used almost all the interactions are reduced. Barring any unforeseen consequences, the use of the last optimization for handling the non-optimized terms significantly improves the distributed MPC performance.

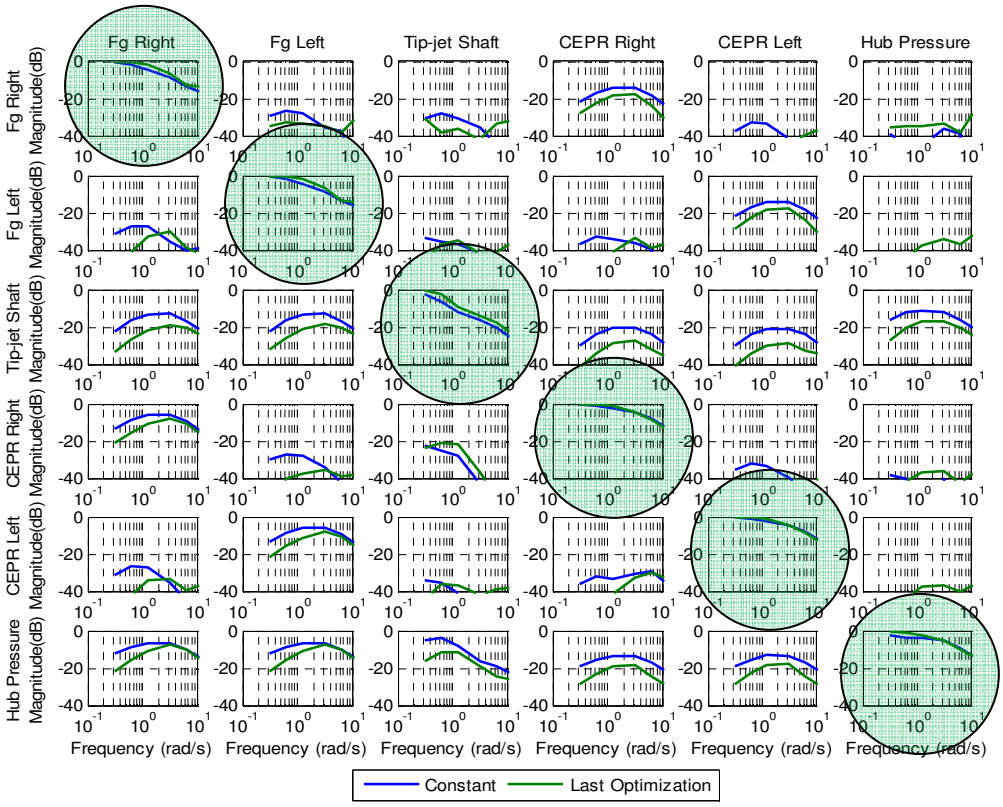

Figure 91: Bode Plot Magnitude Sensitivity to Handling of Non-Optimized Terms

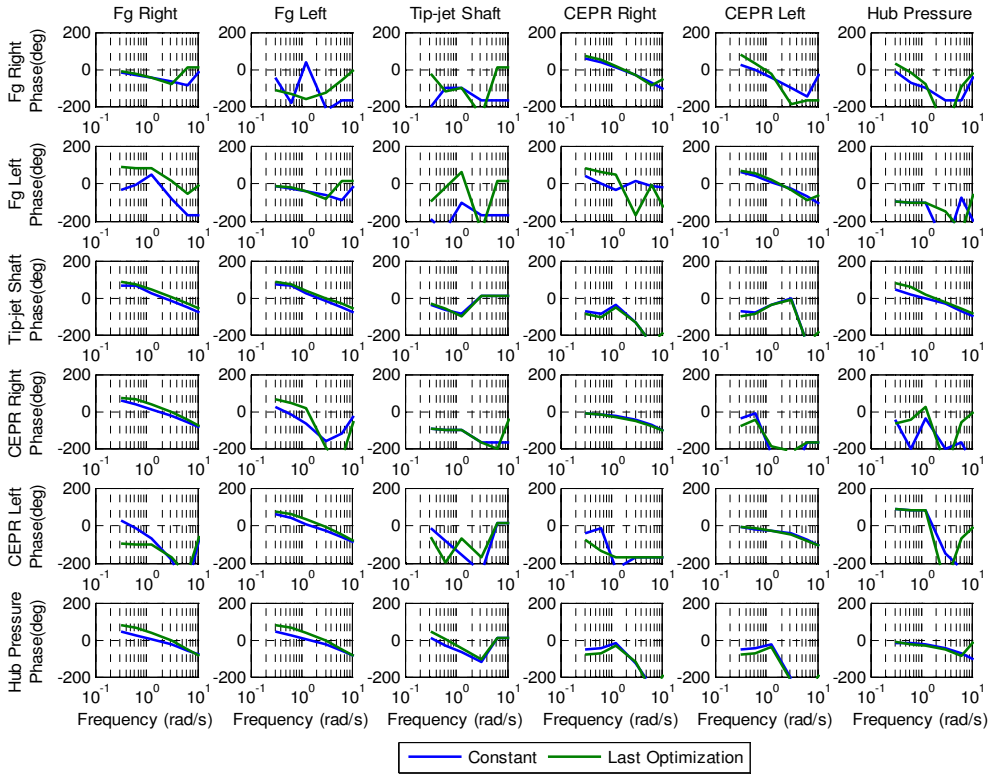

Figure 92: Bode Plot Phase Sensitivity to Handling of Non-Optimized Terms

8.3.1.6 Number of Iterations

Figure 93 and Figure 94 below show Bode plot sensitivity to changes in the number of iterations. The bandwidth and phase margin of the control parameters are not affected by the number of iterations. There is actually a slight increase in the interactions as the iterations occur. Since the computational burden is adversely affected by iteration and there appears to be no significant benefit to more iterations. Therefore, the final choice of numbers of iterations should be one.

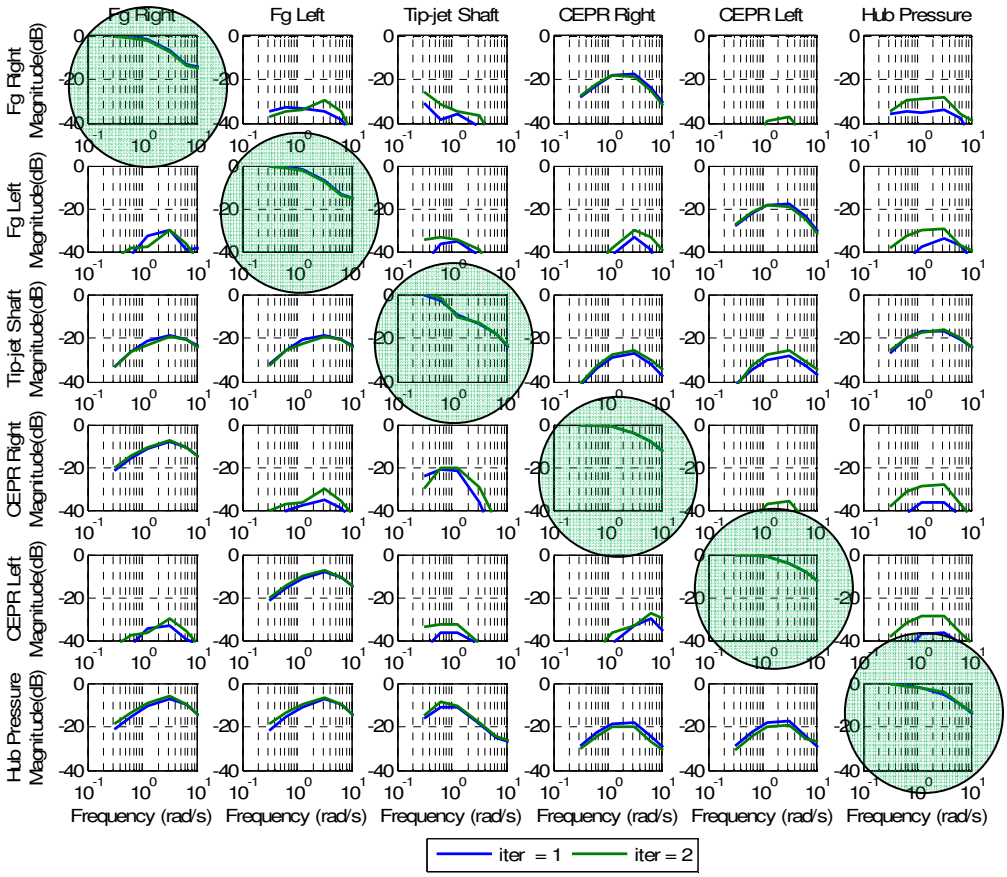

Figure 93: Bode Plot Magnitude Sensitivity to Number of Iterations

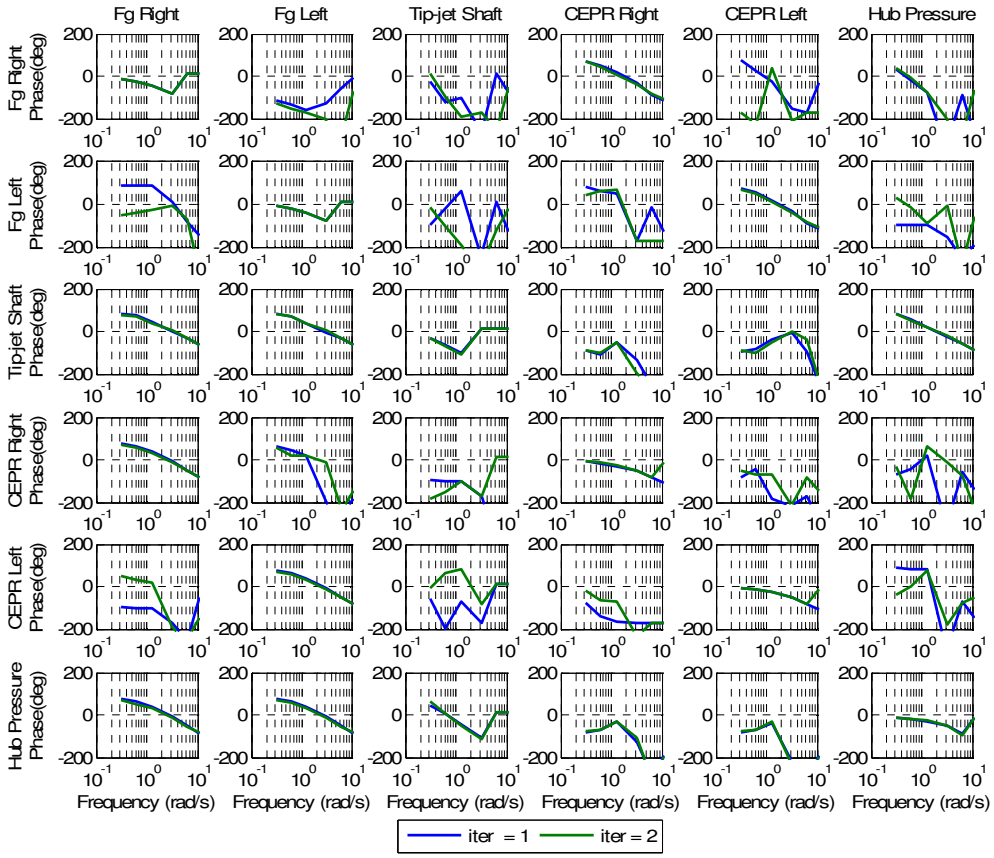

Figure 94: Bode Plot Phase Sensitivity to Number of Iterations

8.3.1.7 Sensitivity Summary

As was seen in the sensitivity study, none of the design points explored satisfactorily met all the control performance targets. Therefore to arrive at a conclusion of the most appropriate choice of design variables some extrapolation of the trends is needed. Some choices of design variables such as control horizon, handling of non-optimized, and number of iterations were obvious and do not need any further discussion.

The final value of the reference tracking weighting function was set to 1, while the control movement weighting function was set to 2. This was a tradeoff between attempting to meet the bandwidth target and minimizing the interactions. For thrust and CEPR the prediction horizon start value was set to 15. Since hub pressure bandwidth was pretty sensitive to changes in H_{we}, hub pressure prediction horizon start value was set to

5. This choice increased the hub pressure interactions with changes in demand; however, it was hoped that the alternating fuel syncing scheme and the handling of non-optimized terms would reduce those interactions. The tip-jet prediction horizon start point was set to 80. Table 28 below summarizes the final design variable choices.

Table 28: Distributed MPC Final Design Variable Choices

Design Variable	Final Value
H_p	100
H_u	20
H_{we}	15
H_{we_hub}	5
H_{wr}	80
Q	1
R	2
w_o	0.1
Fuel	Alternate
Non-Optimized	Last Optimization
Iterations	1

Since the final choice of design variables was not one of the design points explored during the sensitivity study, a Bode plot of this choice is required to understand final performance settings. Below in Figure 95 and Figure 96 show the Bode plot for the final design variables. All the interactions fall below a level of -3 dB. However, there are some interactions that are fairly significant. CEPR interactions with thrust demand change becomes fairly large around the bandwidth frequency of CEPR. Tip-jet shaft speed interactions with changes in thrust and hub pressure demand changes are fairly noticeable around their respective bandwidth frequency. And lastly, hub pressure interactions are fairly noticeable with changes in thrust demand. On a positive note, the large hub pressure interactions with changes in tip-jet shaft speed are significantly reduced. The primary driver for this was the handling of the non-optimized terms.

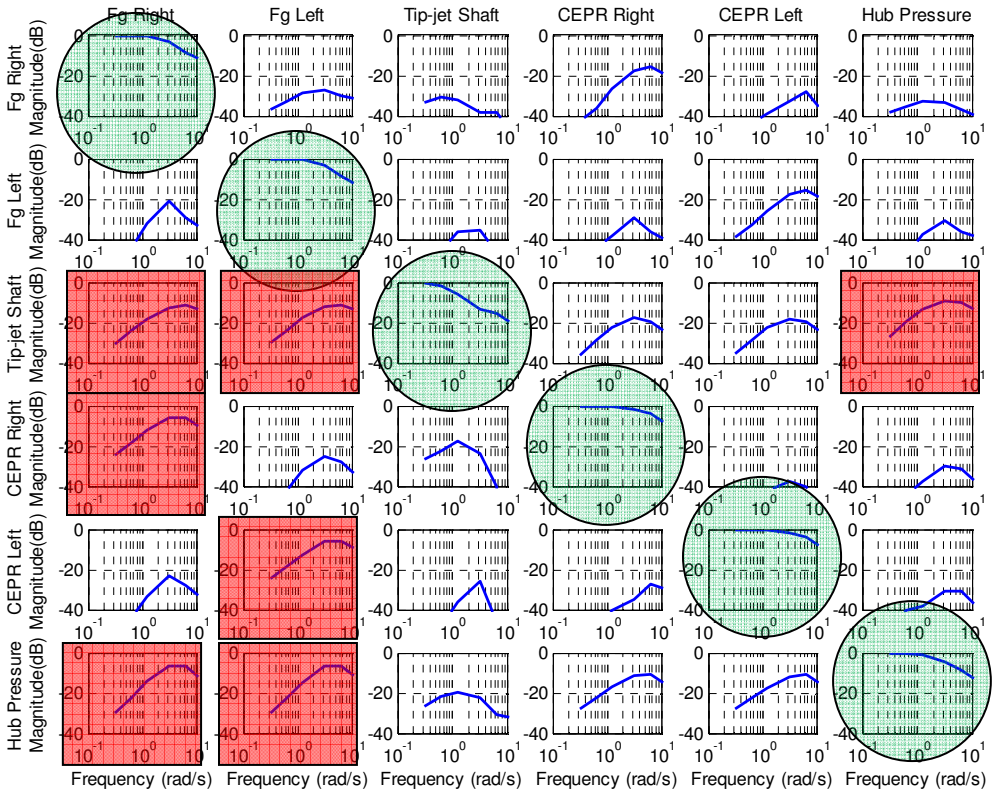

Figure 95: Bode Plot Magnitude of Final Distributed MPC Design

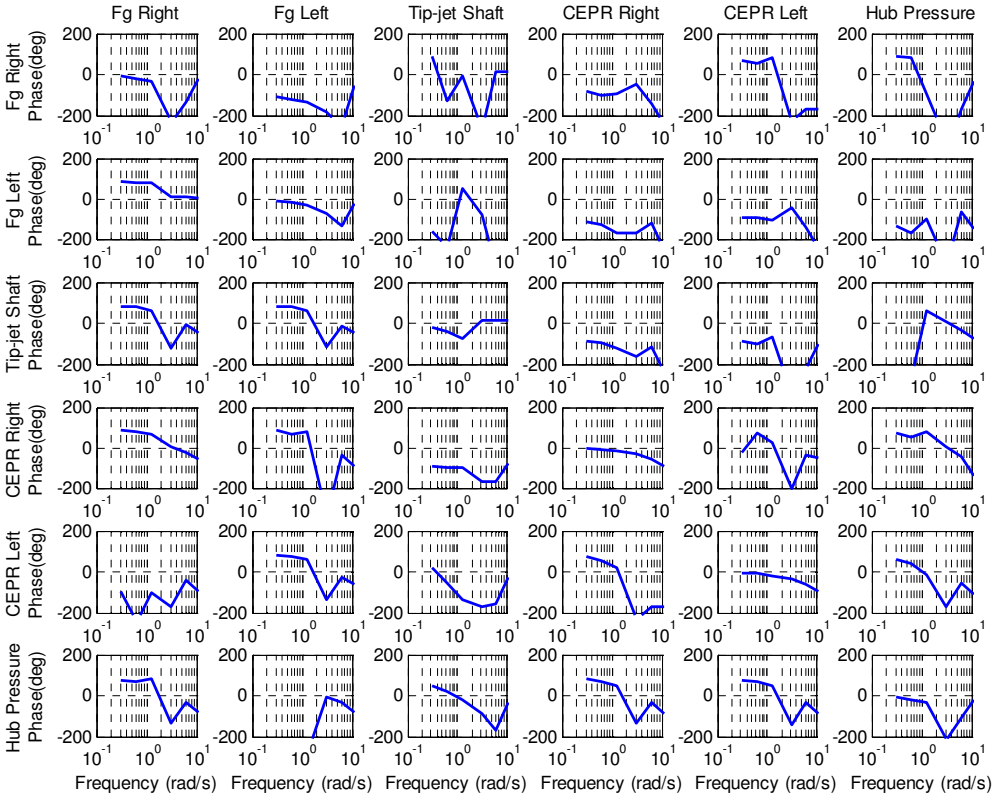

Figure 96: Bode Plot Phase of Final Distributed MPC Design

Table 29 below shows the final control performance metrics for the distributed MPC. Thrust bandwidth almost exactly met the target. Tip-jet shaft speed and CEPR were faster than the target while hub pressure was slightly lower. The bandwidth can fine tuned by adjusted the reference tracking weighting function of each of those variables to meet the targets. Or the integration of an optimizer in the control sizing would drive the parameters closer to the targets. However since there is a large CPU requirement for generating each design point, care should be taken in the optimizer design.

Table 29: Final Distributed MPC Design Control Performance Metrics

Distributed MPC Control Performance	Bandwidth rad/s	Phase Margin degrees
Fg Right	2.9	109
Fg Left	3	109
Tip-jet Shaft	0.82	127
CEPR Right	4.9	130
CEPR Left	4.9	130
Hub Pressure	2.3	42

Lastly, the phase margin of all the variables except hub pressure was significantly greater than the minimal value of 30 degrees. Viewing the phase of hub pressure in Figure 96 shows the trends may be not fully believable and the actual phase margin would be somewhat larger. Running more discrete frequency points would aid in making a much better estimate of phase margin.

8.3.2 Sensitivity to Modeling Error

Identical to the study performed for the centralized MPC, the sensitivity of the control to modeling errors must be accounted for. As mentioned previously, sources of modeling error can come from various aspects of the onboard model modeled wrong or the actual system performing differently than what is captured in the onboard model.

To capture the effect of modeling error on the final design choice of the distributed MPC for the tip-jet reaction drive system, four different modeling error simulations were performed: turbine map error (both efficiency and flow), fan map error (both efficiency and flow), reaction drive duct pressure loss error, and system degradation. For all these simulations, the onboard model is calibrated to the actual system using data matching scalars. To model the turbine map error case, both the turbine efficiency and flow map were scaled by 3%. Similar to turbine modeling error case, the fan map error was modeling by scaling the fan flow and efficiency maps by 3%.

The reaction drive duct pressure loss was simulated by scaling the duct pressure loss by 5%. The degraded system case was modeled by scaling the fan, HPC, HPT, and LPT efficiency and flow maps by factors expected after a significant operational period. The metrics used to compare the candidates was the percent difference between the actual and onboard model estimated non-measured parameters and the percent difference between the actual and target performance metrics. The better candidate should minimize the error between onboard model and the actual system as well as operate at the target parameters. Table 30 summarizes the sensitivity to modeling error for the final design choice. The first case to be analyzed was the turbine map modeling error. For this case, the onboard model overestimates actual engine thrust and hub pressure by about 2.3% and 0.26%, respectively. LP and HP stall margin are overestimated by the model by 9% and 15%, respectively. This fact may result in margining of the stall margin constraint that is used in the control. Turbine inlet temperature is underestimated by about 2.5%.

The onboard model matches the actual system performance much better for the fan map error case. The main reason for this is because both the flow and efficiency map discrepancy for the fan, as opposed to the turbine, can be fully observed using the available sensors and the component data match scalars. All the mode estimated safety and performance parameters match the actual parameters within 0.6% with virtually no error in thrust.

Table 30: Distributed MPC Sensitivity to Modeling Error

	Sensitivity to Modeling Error	Turbine Map Error	Fan Map Error	Reaction Drive Pressure Loss Error	Degraded System
Model-Actual % Difference	Thrust	2.23%	0.00%	-0.60%	0.20%
	LP Stall Margin	9.35%	0.30%	3.90%	2.70%
	HP Stall Margin	14.45%	-0.55%	0.60%	5.30%
	Turbine Inlet Temperature	-2.50%	0.00%	-0.05%	-1.80%
	Hub Pressure	0.26%	-0.21%	-1.70%	0.90%
Target-Actual % Difference	Thrust	2.27%	0.00%	-0.60%	0.20%
	CEPR	0.00%	0.04%	0.01%	0.00%
	Ntip	0.00%	0.00%	0.03%	0.09%
	Hub Pressure	0.26%	-0.20%	-1.70%	0.90%
Data Match Scalars	Fan Eta	1.002	0.967	0.999	0.980
	Fan Wc	0.996	0.971	1.042	0.979
	HPC Eta	0.993	1.001	0.998	0.979
	HPC Wc	0.989	1.001	0.997	0.978
	HPT Wc	1.003	1.000	1.000	1.013

Similarly to the turbine fault, when the reaction drive duct pressure loss model is wrong, the model estimated parameters differ from the actual parameters. However the main differences appear in the LP stall margin and hub pressure estimates which vary by 3.9% and 0.7%, respectively. When the pressure loss in the duct is underestimated, the actual hub pressure is higher than the estimated pressure resulting in a significantly lower steady state fan stall margin. This fact makes highlights the importance of having a good rotor pressure loss model.

For the degradation case, the onboard model matches actual thrust by within 0.2%. However, both stall margin and turbine inlet temperature differ from the actual system by at least 2.5%. In this case the hub pressure is over estimated by around 1% resulting in an increase in steady state fan stall margin.

8.3.3 Disturbance Rejection

In addition to good control input to control output response, it is important the controlled parameters do not deviate significantly from the demanded values when faced with a change in rotor load. Below in Figure 97 the responses of the MPC to various different sinusoidal inputs of the rotor load. In contrast to both the PI controller and the centralized all control parameters are at a similar order of magnitude, with the peaks corresponding to the parameter's bandwidth. However, the largest peak magnitude of the disturbance was about -15 dB, which is much below the bandwidth target of -3 dB. Therefore from this analysis, the control system should be able to suitable reject changes in the rotor load.

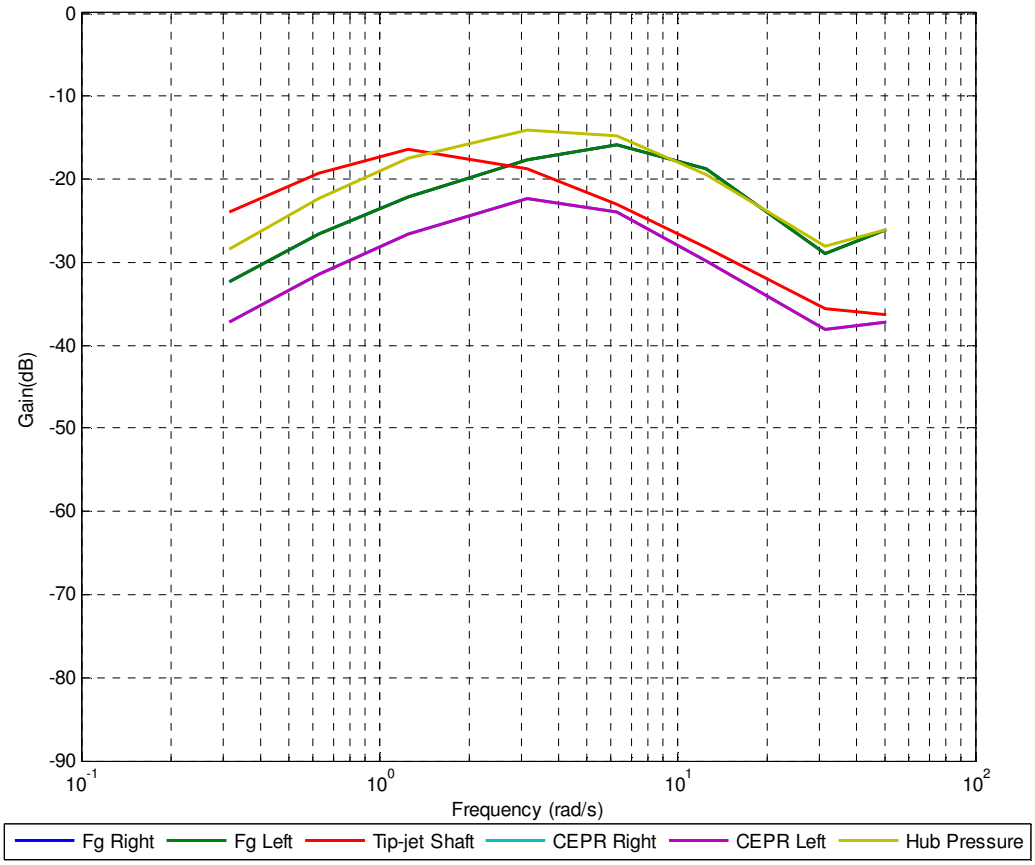

Figure 97: Distributed MPC Response to Rotor Load Disturbance

8.4 **Distributed MPC Simulations**

The purpose of the simulations in the next few sections is to provide an understanding of the overall performance/response of the distributed MPC. Specific design variables such as varying throttle settings, disturbance rejection, limit avoidance, and robustness to degradation are explored in detail.

8.4.1 Rotor Load and Throttle Demand Change Simulations

The following sections are broken up into analysis of engine throttle changes, rotor load demand changes, and both engine throttle and rotor load demand changes. The engine throttle changes provide an opportunity to analyze how the control responds when multiple demands are changed simultaneously. For a tip-jet reaction drive system, a change in rotor load demand provides a natural opportunity to analyze disturbance rejection. And the final section will combine the two types of demand changes to see if there are any additional observations.

8.4.1.1 Throttle Demand Changes

The first transient to analyze is a change in engine throttle demand. As the engine throttle demand changes, the engine thrust demand changes. The CEPR demand also changes with throttle demand to maintain close to design compressor stall margin. Although physically associated with the tip-jet subsystem, the hub pressure demand changes with throttle changes because of its strong correlation with fan stall margin. The demand schedules were defined and developed while the cycle was operating at new and clean conditions (i.e., component performance matched component map performance).

Figure 98 and Figure 99 below shows a step change in throttle demand for both a new and clean system. In all the transients, it takes the approximately two seconds for the engine thrust and hub pressure to settle to the new demand values. CEPR has a noticeable overshoot in the throttle decrease. Assuming the control can handle stall

margin suitable, this overshoot should not be to concerning. From the perspective of the tip-jet subsystem, the tip-jet fuel flow is increased to generate more torque (as seen through an increase in velocity) in the tip-jet rotor to overcome the decrease in airflow and source pressure to the tip-jet subsystem. The distributed MPC response is very similar to centralized MPC.

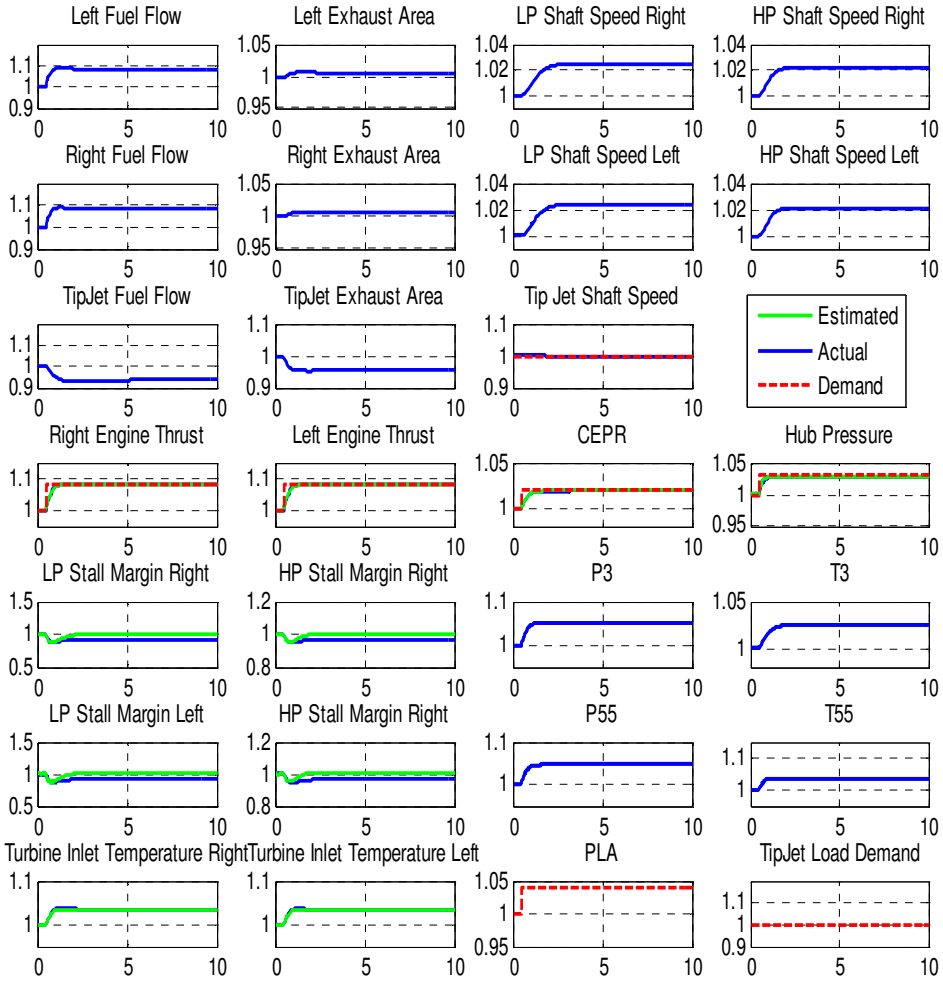

Figure 98: Engine Throttle Demand Decrease for a New and Clean System

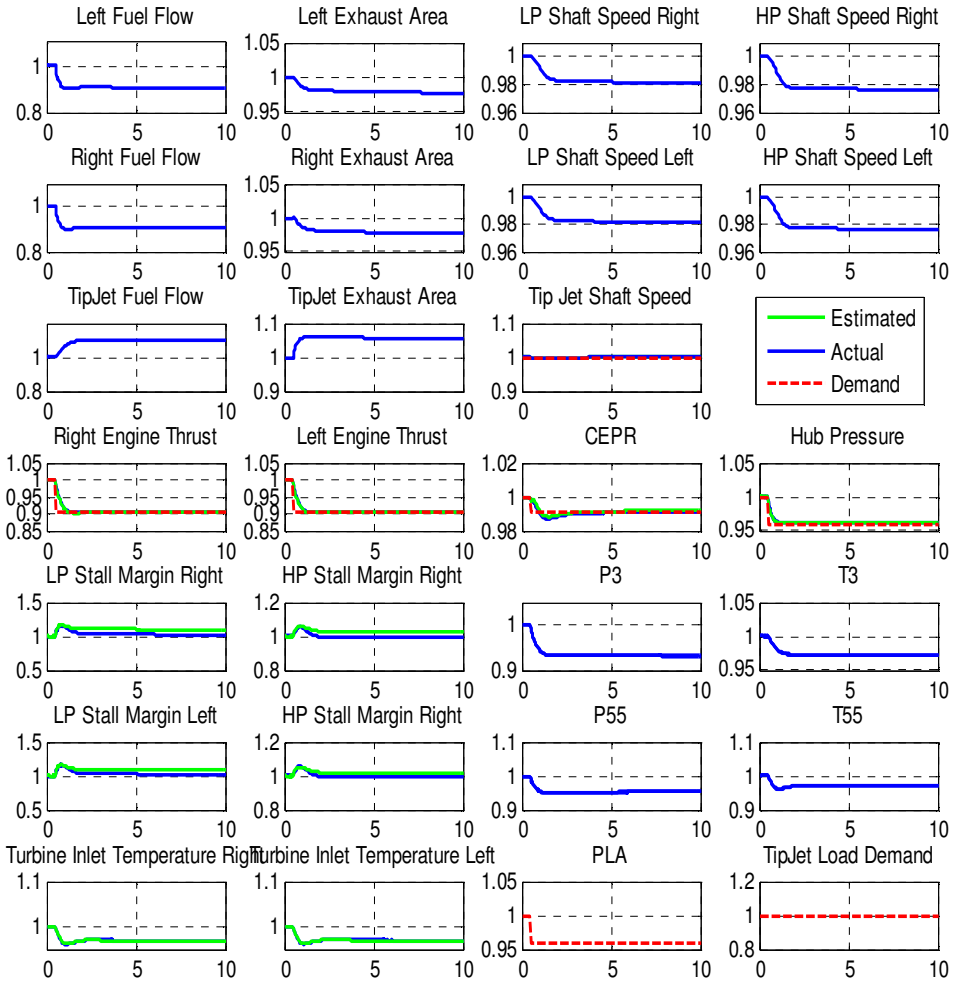

Figure 99: Engine Throttle Demand Increase for a New and Clean System

Figure 100 and Figure 101 below show a step change in throttle demand for a degraded system. As expected the temperatures in for the degraded system are elevated relative to their new and clean counterparts. Besides the elevated temperatures, the response of the degraded system is almost identical to that of the new and clean system.

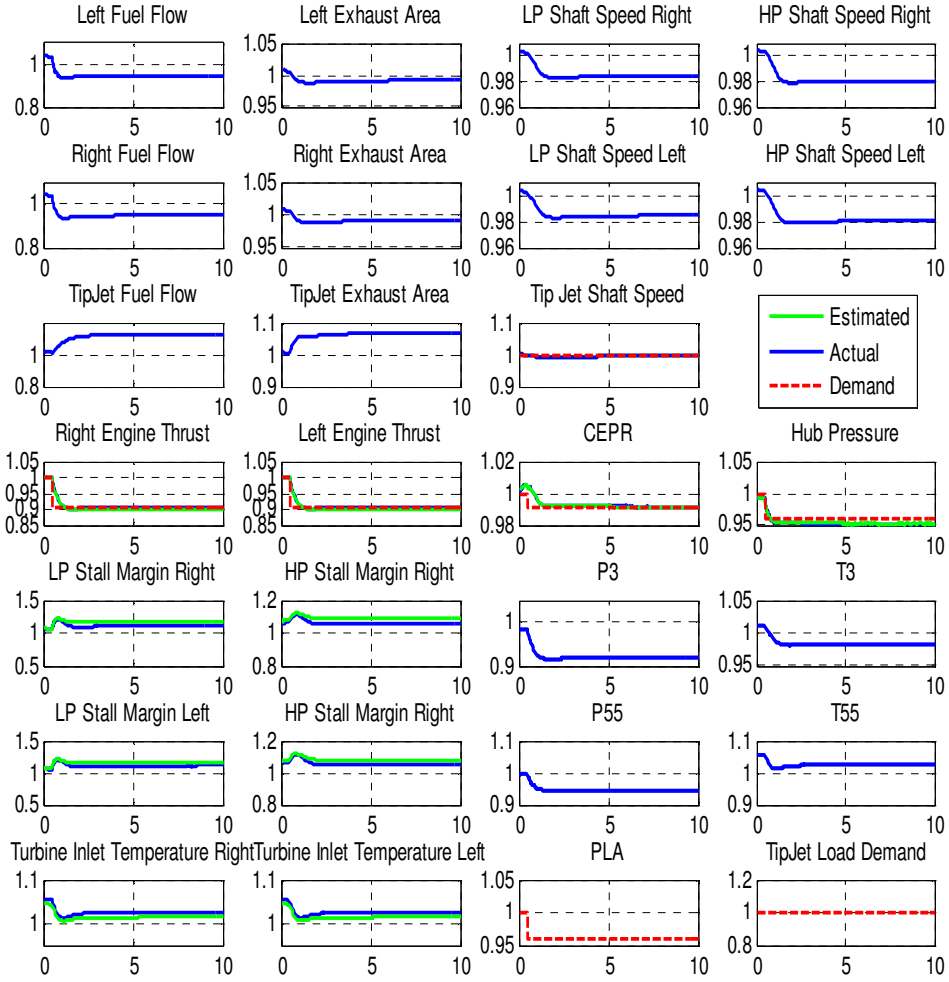

Figure 100: Engine Throttle Demand Decrease for a Degraded System

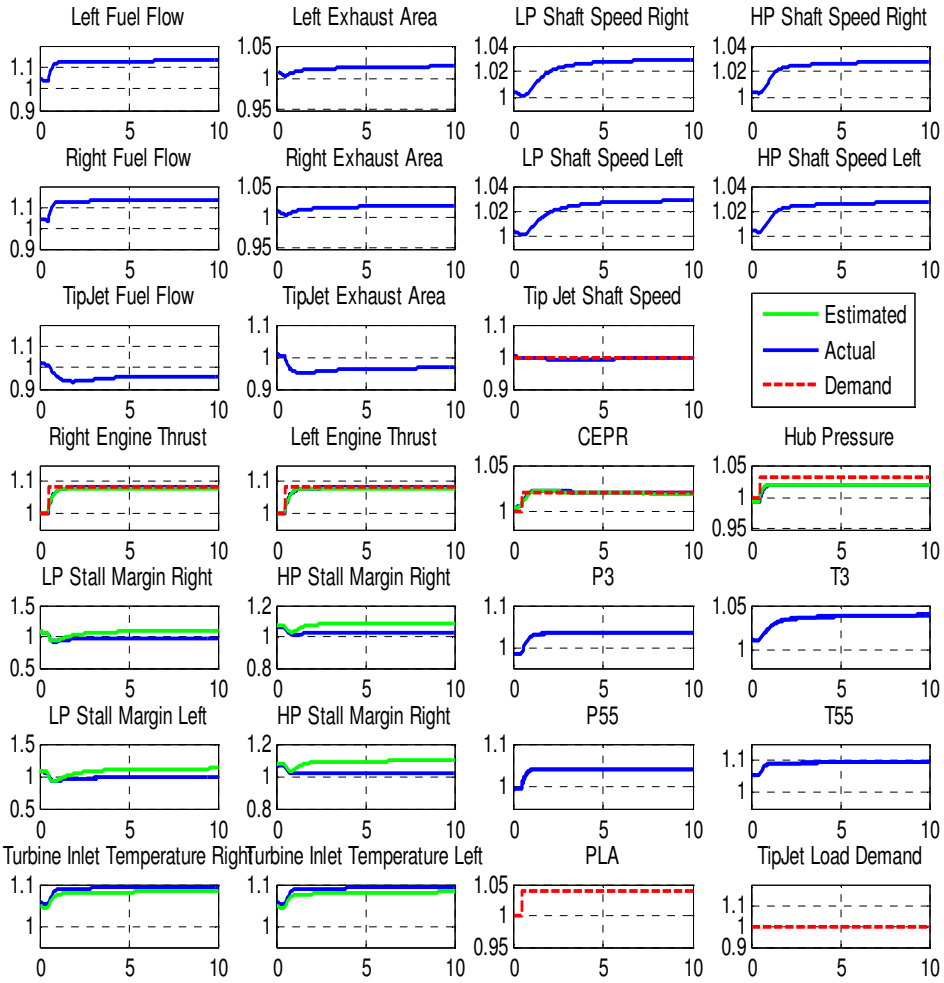

Figure 101: Engine Throttle Demand Increase for a Degraded System

Throughout all the transients there is a noticeable difference between the stall margins estimated by the onboard model and the actual stall margin. In general it appears that the onboard model typically overestimates the available stall margin. To overcome this limitation, the minimum amount of stall margin used in the constraint may need to be margined to ensure the system doesn't stall because of model uncertainty.

For the new and clean system, the onboard model provides a fairly accurate estimate of the turbine inlet temperature. However as the system degrades, the onboard model underestimates turbine inlet temperature by about 1%. To account for this in the

constraints, the constraints should be margined by about 1% to ensure the actual turbine inlet temperature does not exceed any design values, thus reducing the available life of the system.

8.4.1.2 Rotor Load Demand Changes

Another transient of interest for the tip-jet reaction drive system is the response to a change in tip-jet rotor load. This load change can be induced from either direct pilot input via a change in the collective or indirectly from changes in ambient conditions, such as a wind gust. From the perspective of the tip-jet subsystem control, or more generally the overall control, a rotor load change is viewed as a disturbance. Since it is a disturbance, none of the speed or pressure demands change when a rotor load change is encountered. Similarly to the centralized MPC, the distributed MPC views changes in the rotor load as measured disturbances, where the effect of changes in the load on the system is known.

Figure 102, Figure 103, Figure 104, and Figure 105 below show rotor load increase and decreases on both new and clean and degraded systems. Similarly to the throttle change simulations, the system response for all four scenarios is very similar. For all the transients, the tip-jet rotor shaft speed takes around 5 seconds to settle after exposed to a 10% change in rotor load. With the only significant differences between the new and clean and degraded system cases are the elevated speeds and temperatures of the degraded system. This is similar to the settling time seen for the tip-jet shaft for the throttle change simulations. A 1% perturbation is noticeable in CEPR and hub pressure. Fine tuning of the design variables such as reference tracking weighting factor may reduce the amount of interactions in CEPR. There does not appear to be any significant effect of the engine thrust to changes in the rotor load. As with the engine throttle simulations, these simulations appear very similar to the centralized MPC.

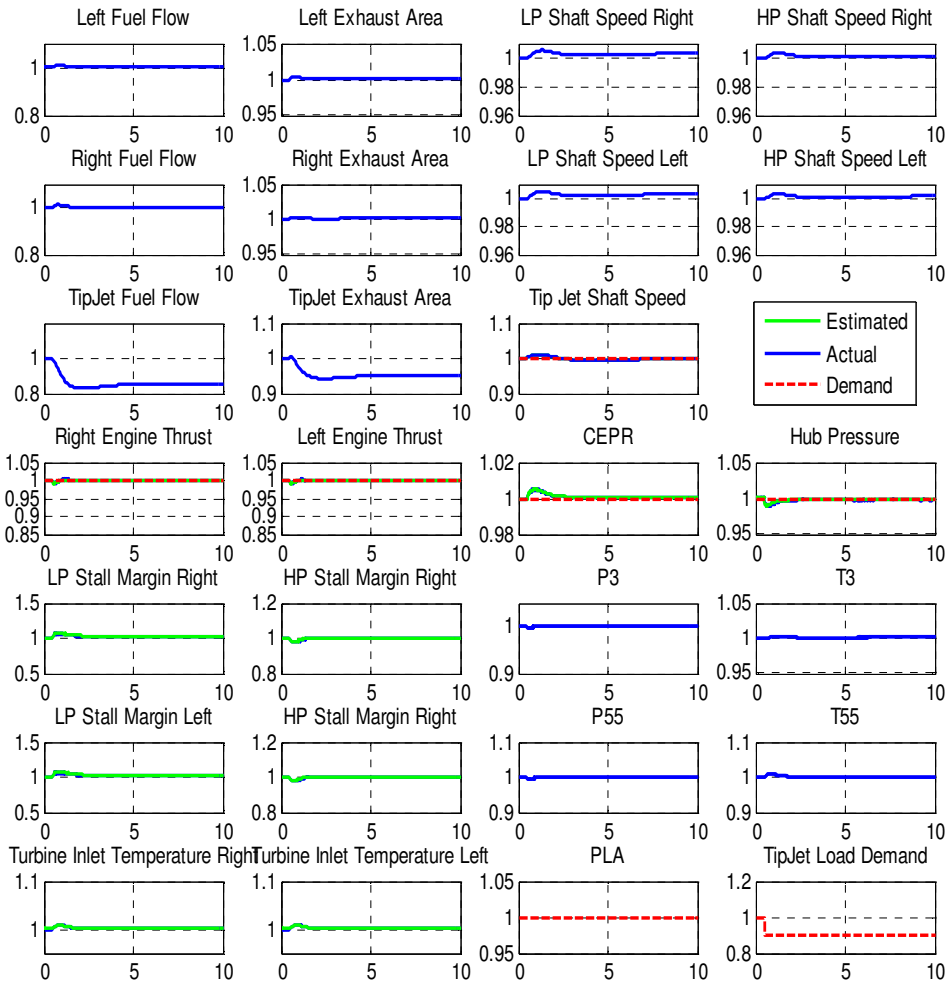

Figure 102: Rotor Load Decrease for a New and Clean System

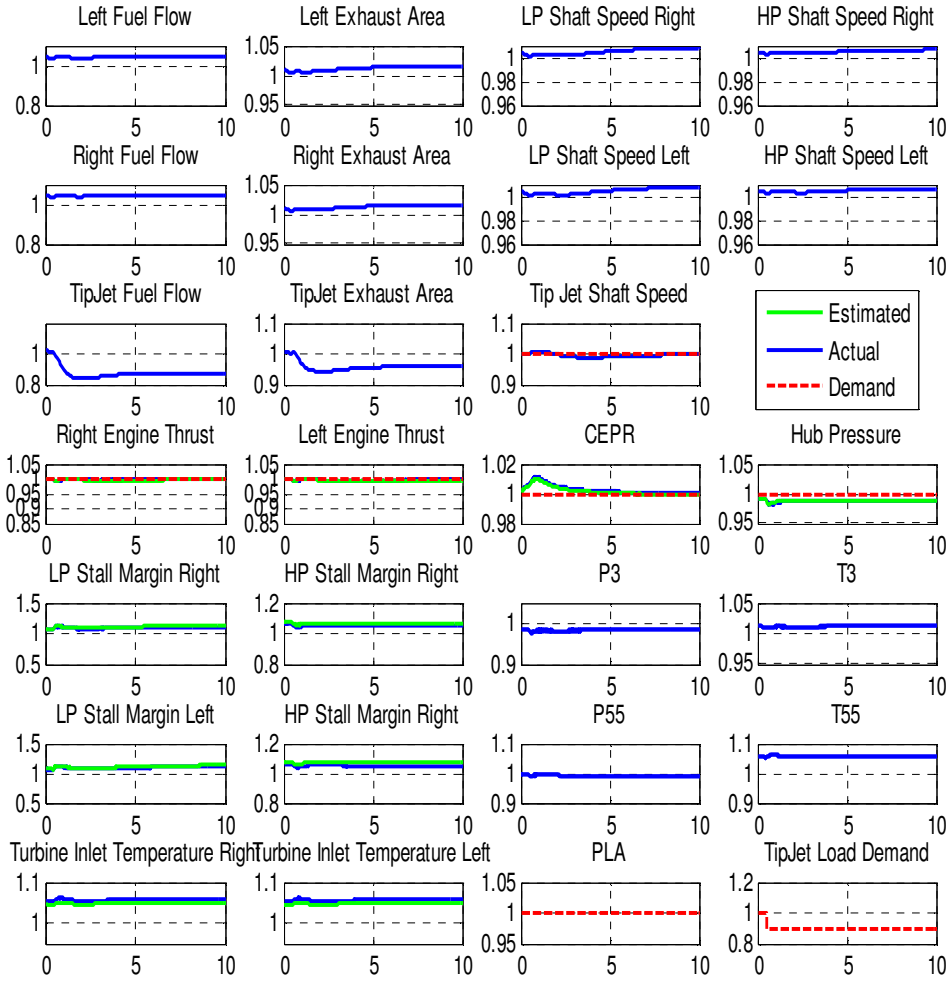

Figure 103: Rotor Load Decrease for a Degraded System

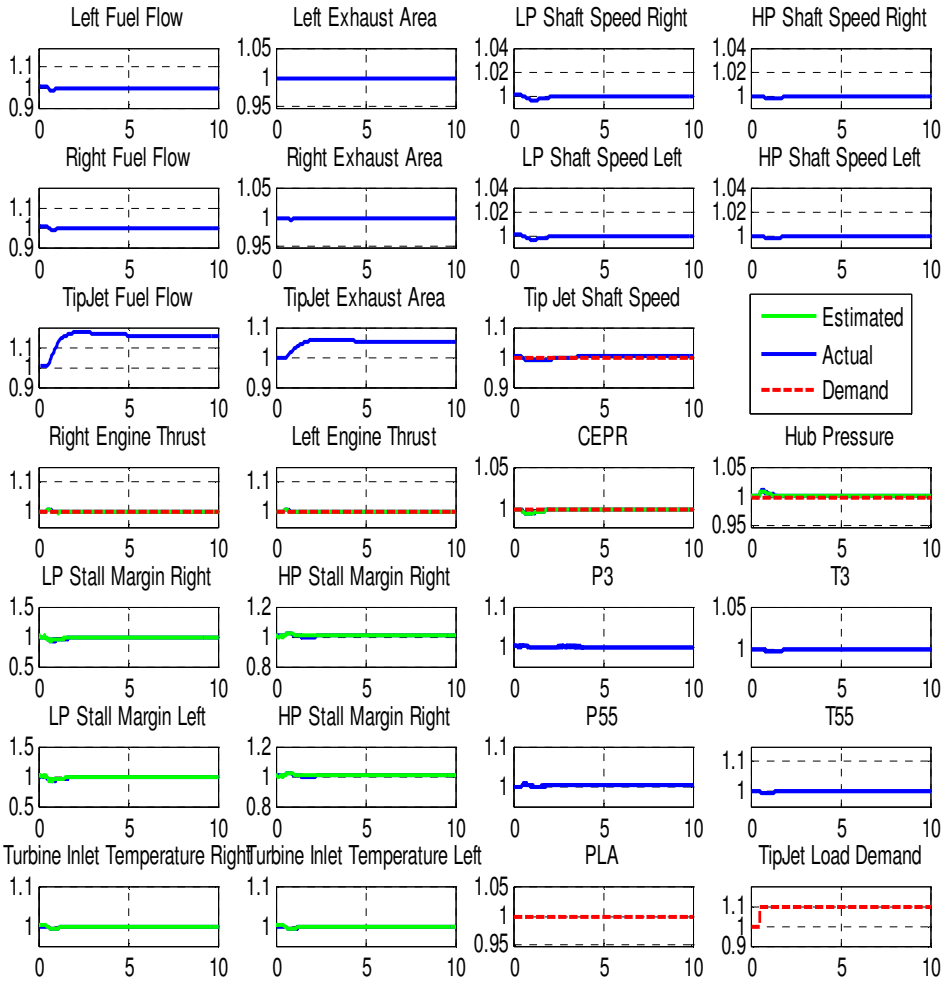

Figure 104: Rotor Load Increase for a New and Clean System

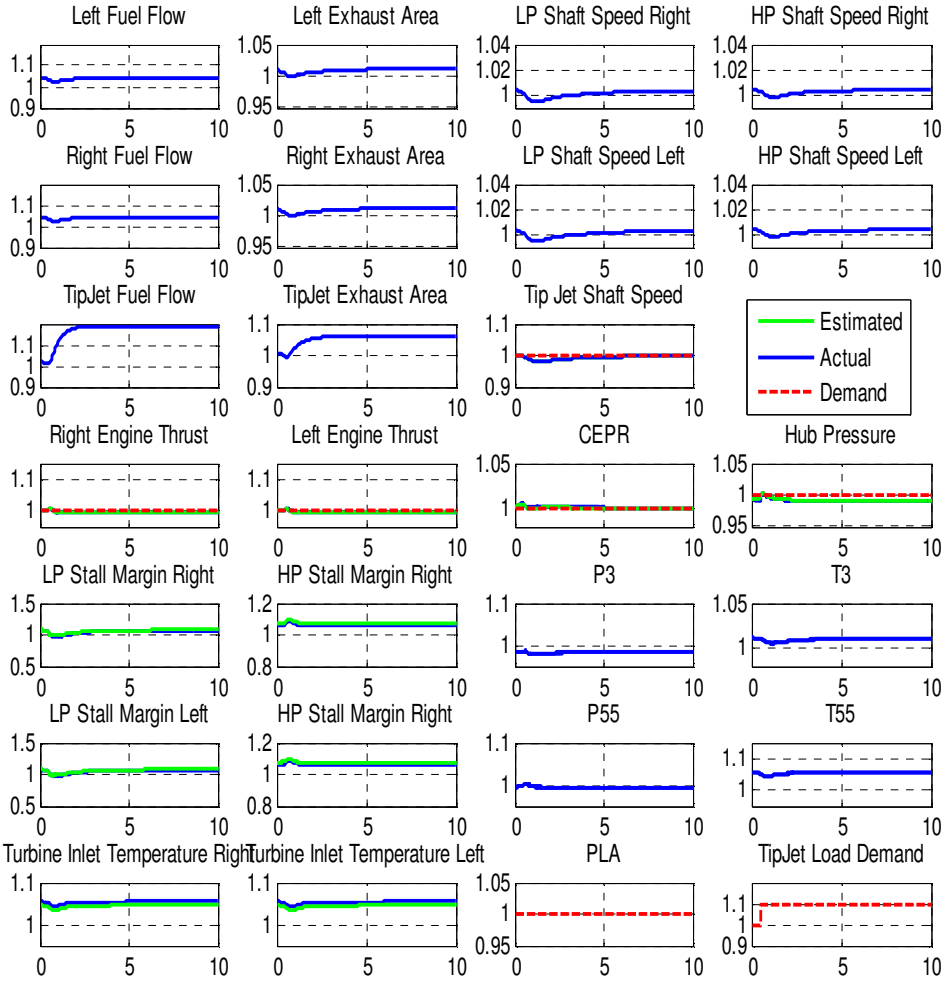

Figure 105: Rotor Load Increase for a Degraded System

8.4.1.3 Throttle and Rotor Load Changes

The last transient to analyze combines the two previous transients. Figure 106,

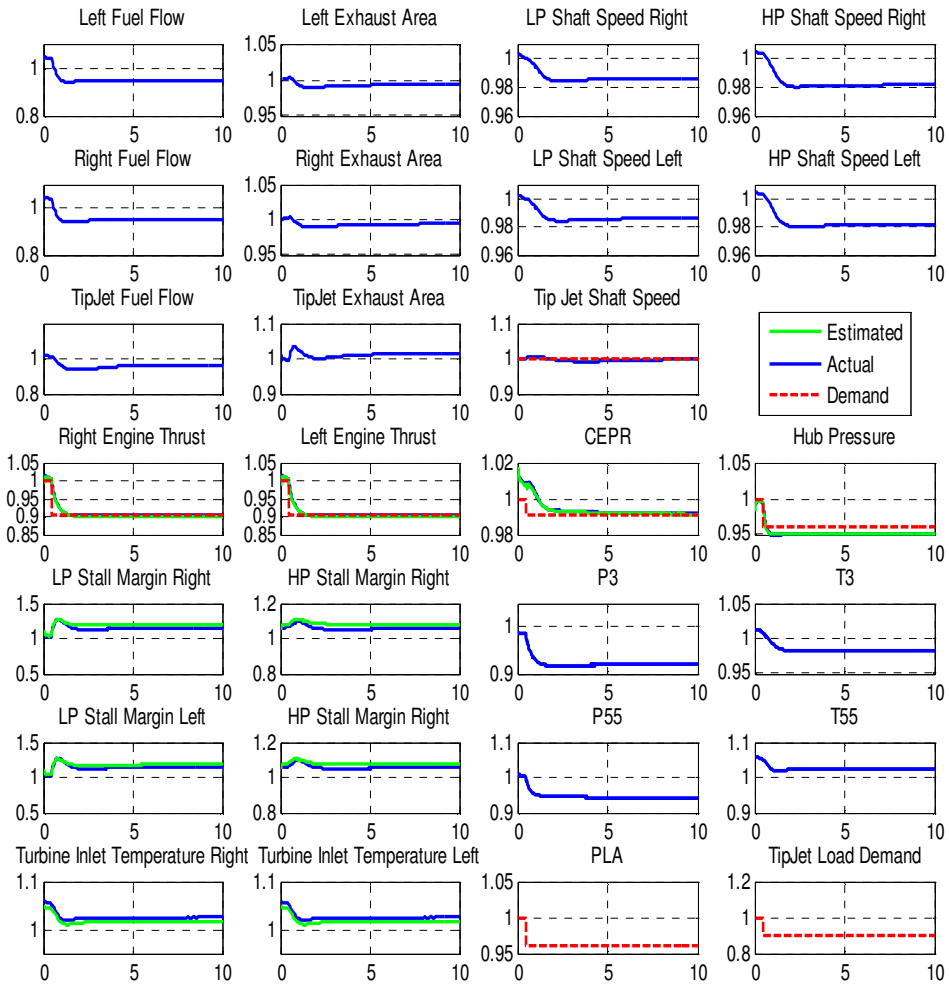

Figure 107, Figure 108, and Figure 109 below show rotor load/engine throttle decreases on both new and clean and degraded systems. Matching the previous MPC simulations, the tip-jet rotor speed settles approximately 5 seconds after the initialization. And the other demanded parameters settle on the order of 2 seconds after the transient starts. h CEPR has the overshoots seen in the previous simulations. In the general the trends in these simulations are similar to the trends seen in the previous runs. There do not appear

to be any additional interactions or effects resulting from the combination of the two demand changes; and the conclusions drawn from the previous sections can be applied to the analysis of these transients.

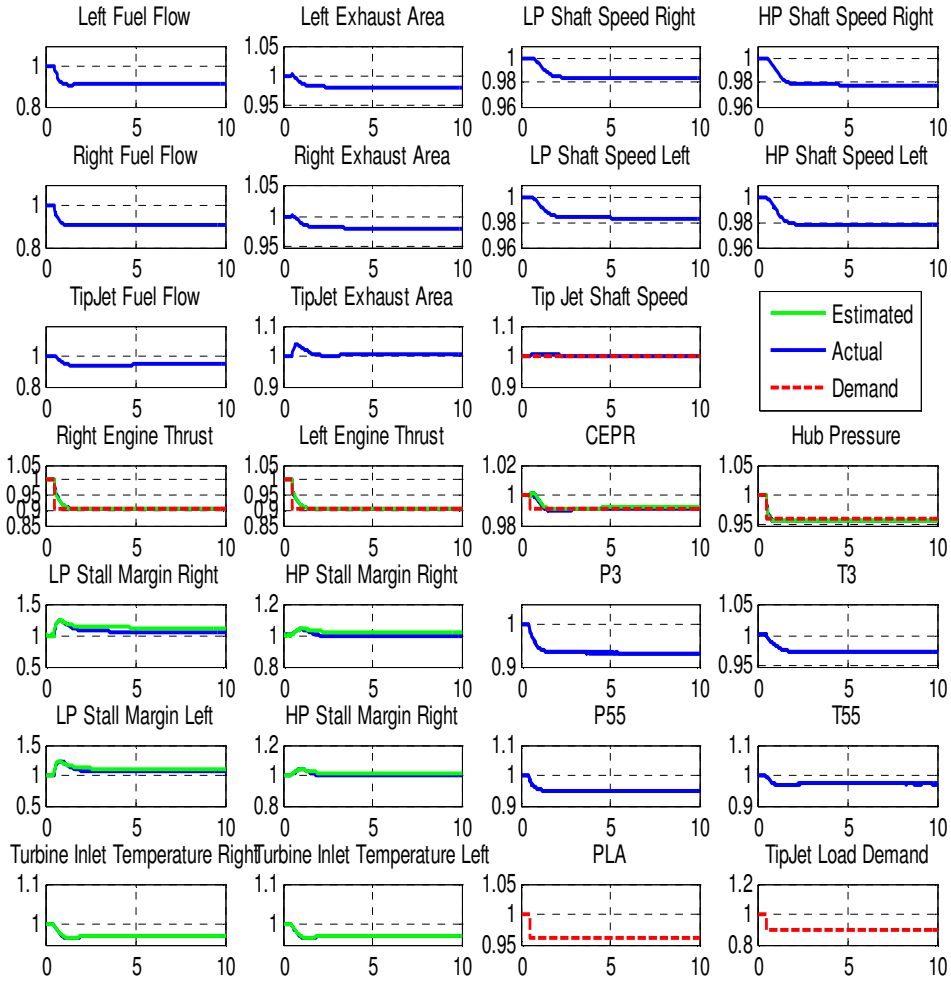

Figure 106: Engine Throttle and Rotor Load Decrease for a New and Clean System

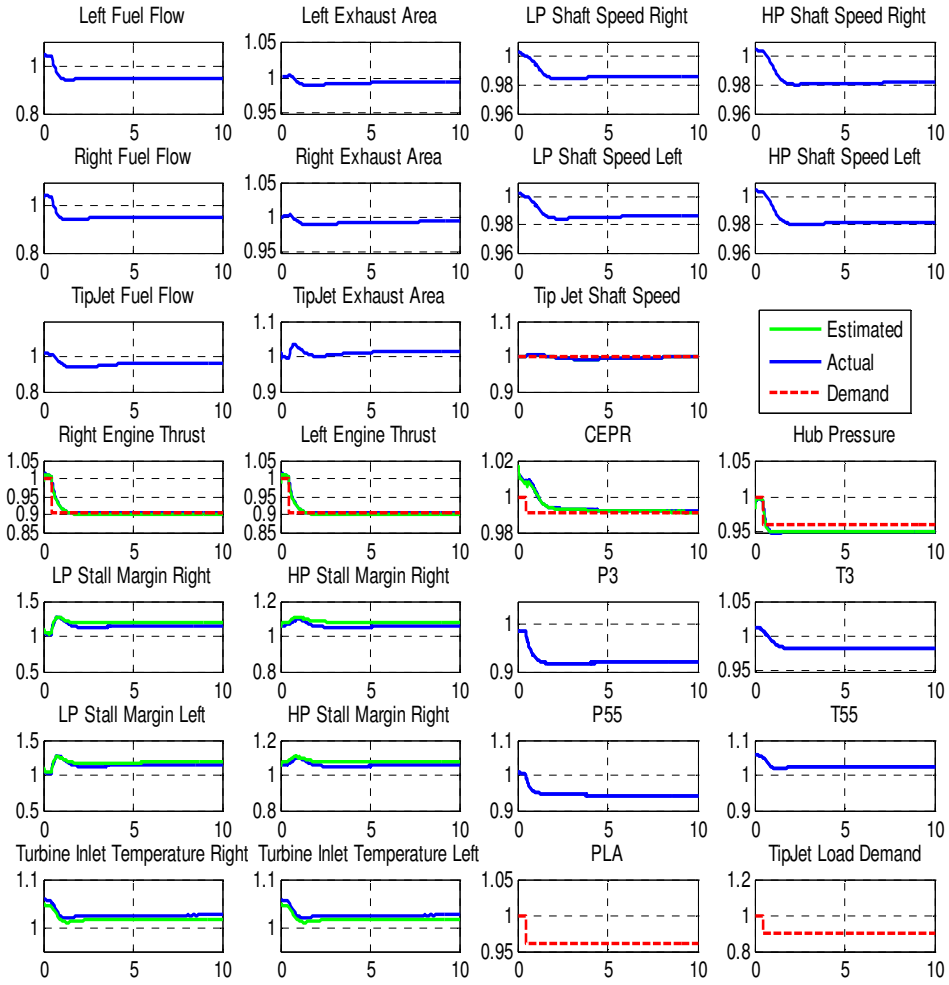

Figure 107: Engine Throttle and Rotor Load Decrease for a Degraded System

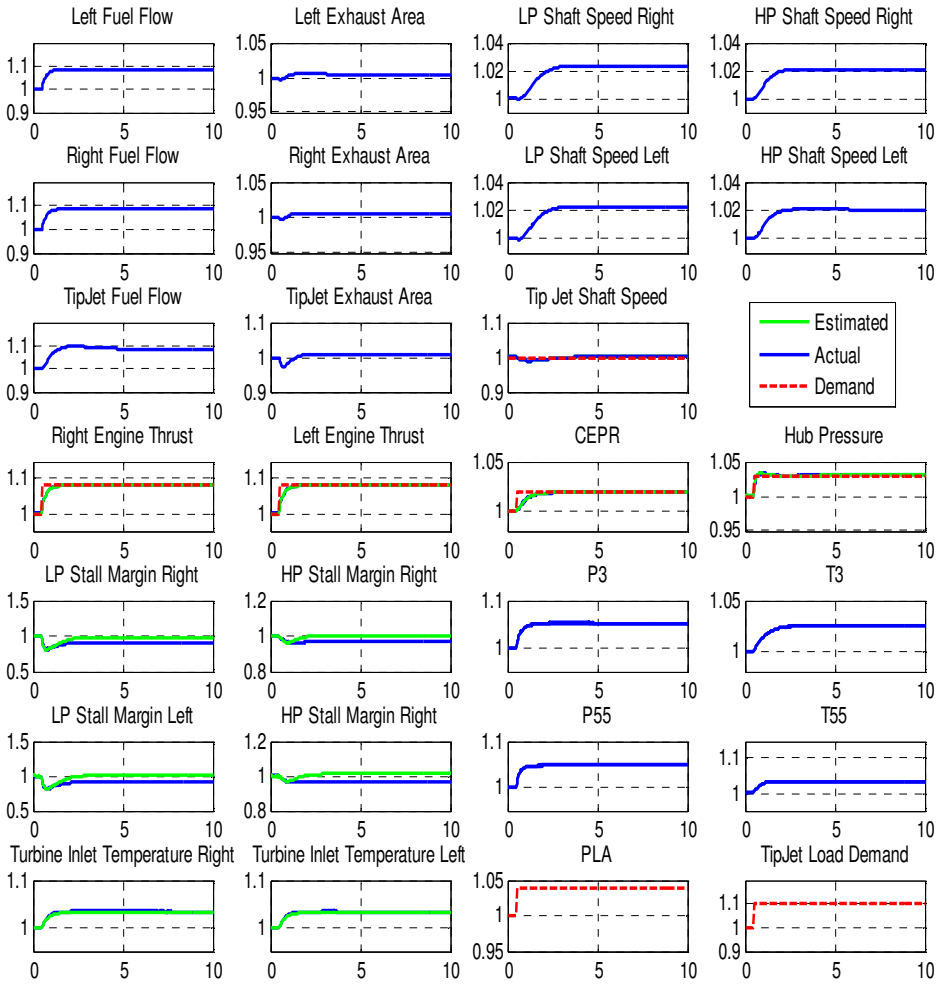

Figure 108: Engine Throttle and Rotor Load Increase for a New and Clean System

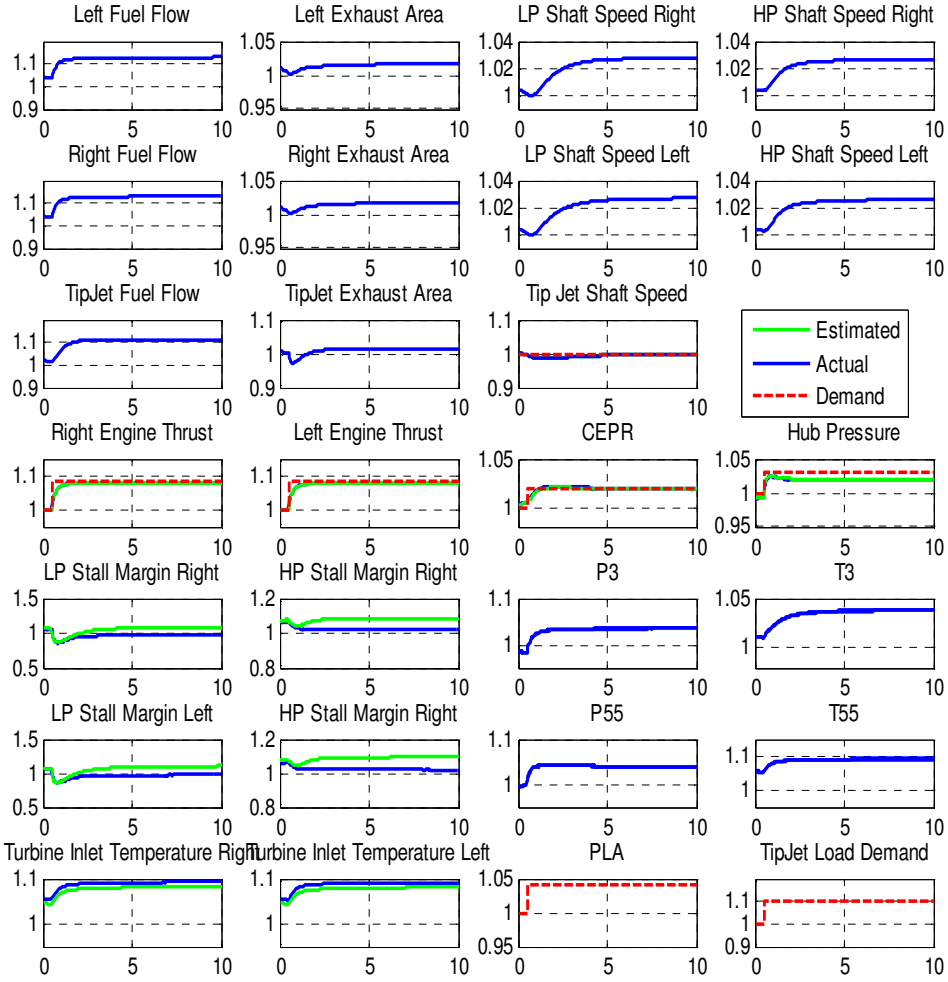

Figure 109: Engine Throttle and Rotor Load Increase for a Degraded System

8.4.2 Constraint Handling

As has been mentioned throughout, a primary benefit of MPC is the handling of constraints. The use of constrained optimization techniques allow the integration of constraints handling and reference tracking into a single control framework. MPC can handle constraints on measured output, unmeasured output, control input position, and control input rate change. This section is going to demonstrate the constraint handling for two cases of interest: stall margin limit avoidance and control input rate limitations.

To handle constraints in the centralized MPC all the constraints can be included in the control. However, more care must be taken to appropriately handle constraints for the distributed MPC. If all the constraints were included in all the controllers, the optimization problem would be ill conditioned and resultant answer would be wrong. Therefore at each time-step of the controller, the only constraints that should be active should be the constraints that can be controlled by the variable being optimized. As with the control input/output selection, the relative gain array can be used to help identify which constraints should be associated with which control inputs.

Table 31 and Table 32 below summarize the sensitivity of stall margin and turbine inlet temperature to each of the control inputs. Engine fuel flow can be used to control HP stall margin and both turbine inlet temperatures limits. While engine exhaust area can be used to control both stall margins and turbine inlet temperatures. Tip-jet fuel flow does not have a significant effect on either stall margin or turbine inlet temperatures. While tip-jet exhaust area can also be used to control fan stall margin.

Table 31: Stall Margin Sensitivity to Control Inputs

Stall Margin Constraint Relative Gain Array	Right Fuel Flow	Right Exhaust Area	Left Fuel Flow	Left Exhaust Area	Tip-Jet Fuel Flow	Tip-Jet Exhaust Area
Fan Stall Right	0.0822	0.2651	0.0872	0.0706	0.0455	0.4493
HPC Stall Right	0.444	0.5283	0.0147	0.011	0.0022	-0.0003
Fan Stall Left	0.0873	0.0705	0.0824	0.265	0.0455	0.4493
HPC Stall Left	0.0146	0.0112	0.4434	0.5288	0.002	-0.0001

Table 32: Turbine Inlet Temperature Sensitivity to Control Inputs

Turbine Inlet Temperature Relative Gain Array	Right Fuel Flow	Right Exhaust Area	Left Fuel Flow	Left Exhaust Area	Tip-Jet Fuel Flow	Tip-Jet Exhaust Area
HPT Inlet Temperature Right	4.136	-1.9138	-0.2305	0.0977	-0.0944	-0.995
LPT Inlet Temperature Right	-3.3159	2.8945	0.2398	-0.1027	0.1138	1.1705
HPT Inlet Temperature Left	-0.2296	0.0973	4.1319	-1.9096	-0.0892	-1.0008
HPT Inlet Temperature Left	0.2389	-0.1024	-3.3117	2.8903	0.1085	1.1765

8.4.2.1 Engine Stall Margin

To demonstrate the distributed MPC stall handing capability simulations were run where the hub pressure demand increased while holding all other demands constants. In this case the fan pressure ratio will increase resulting in the system operating at a lower stall margin steady-state fan stall margin. Figure 110 and Figure 111 below shows plots of both the unconstrained and constrained simulations. The former is unconstrained and the latter has a constraint on both the LP and HP stall margin. For the purpose of these simulations, the stall margin was constrained to be no lower than 75% of the design value. In reality this value would be much lower, however to ensure the nonlinear system model converges in the unconstrained case, the stall margin limit was defined at such a high level. In the constrained simulation, the LP stall margin constraint becomes active resulting in the system oscillating between a lower CEPR and hub pressure. This appears to be a drawback of the distributed MPC.

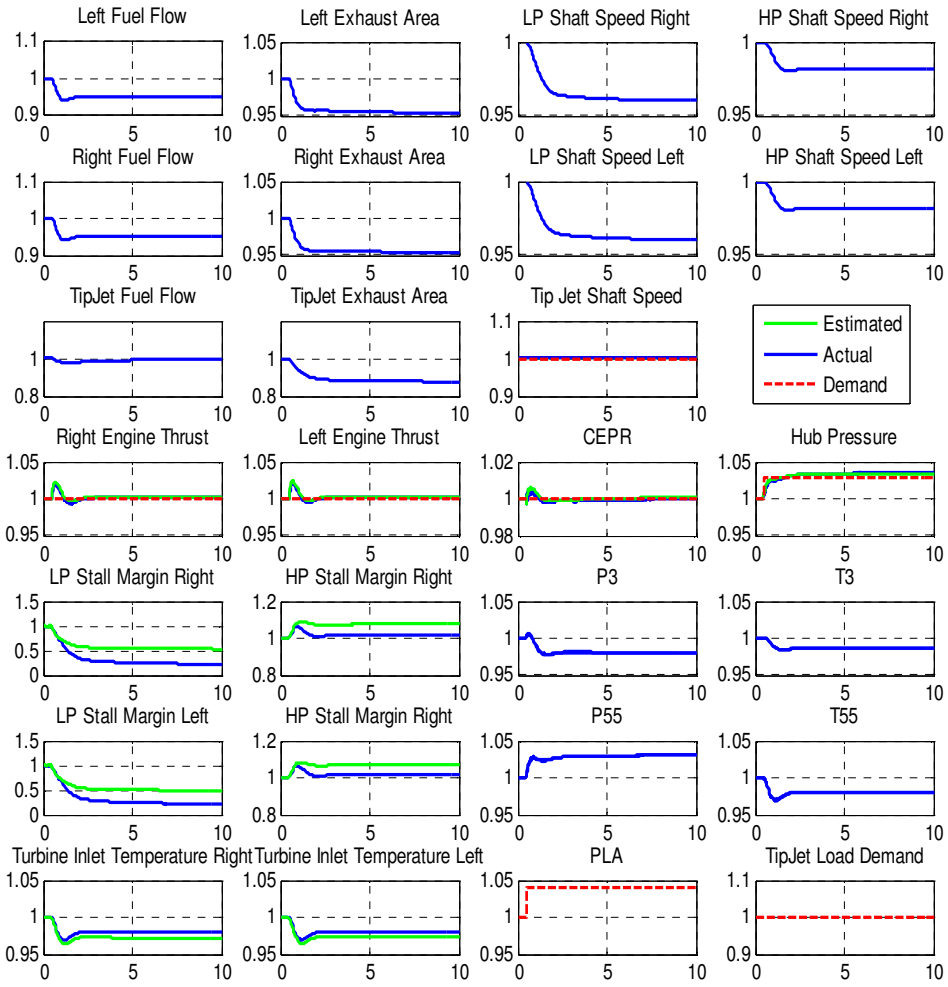

Figure 110: Unconstrained Hub Pressure Increase

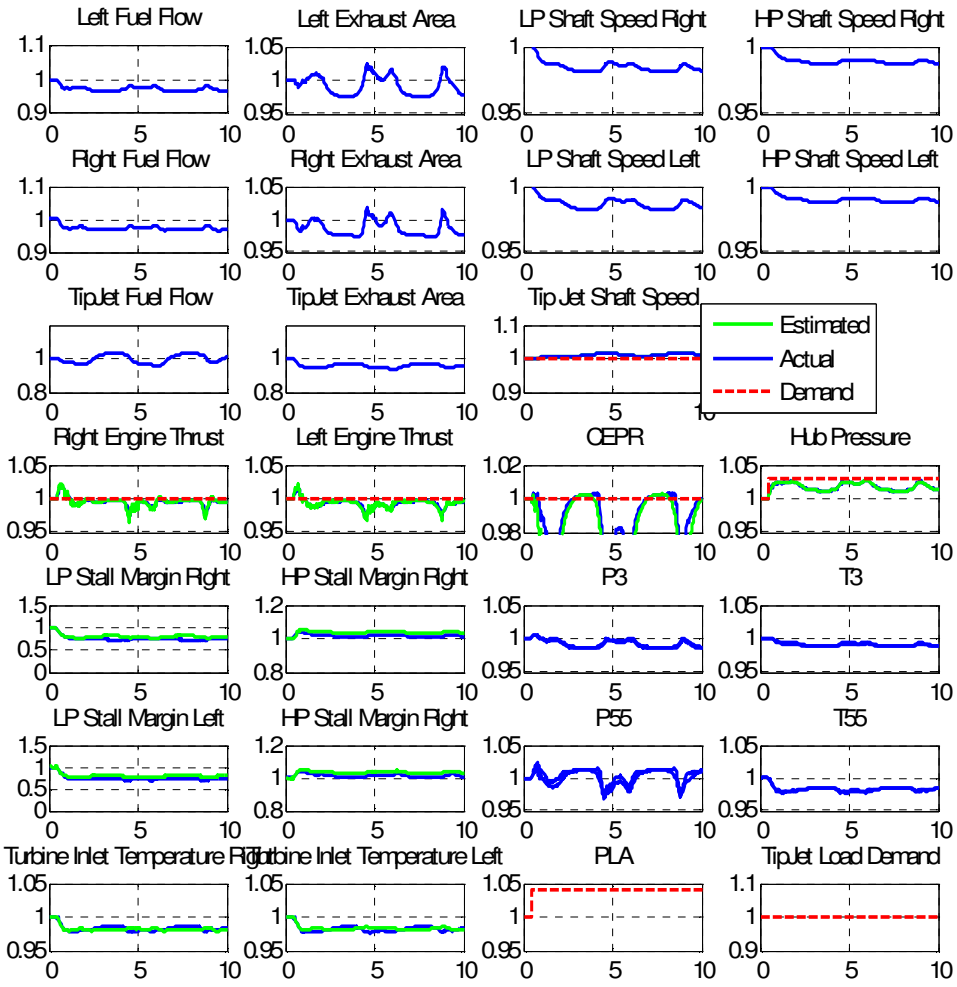

Figure 111: Constrained Hub Pressure Increase

In addition to LP stall margin, there are significant differences between the unconstrained and constrained CEPR, hub pressure, and thrust response. Figure 112 below compares some of the control parameters cases. The sharp movements in all the parameters appear to be mirroring the trends in exhaust area. An initial diagnosis was because of the large tip-jet shaft speed interaction with hub pressure. This could be alleviated by changing the design variables (such as decreasing reference tracking weighting) to lower that interaction. Although tip-jet fuel flow did not significantly

affect LP stall margin, the inclusion of a LP stall constraint when tip-jet fuel flow was being optimized delayed the onset of the oscillations. Combining these two observations may help eliminate the oscillations. However, even though there is significant oscillation in multiple parameters, the system does not violate the constraint and operates safely.

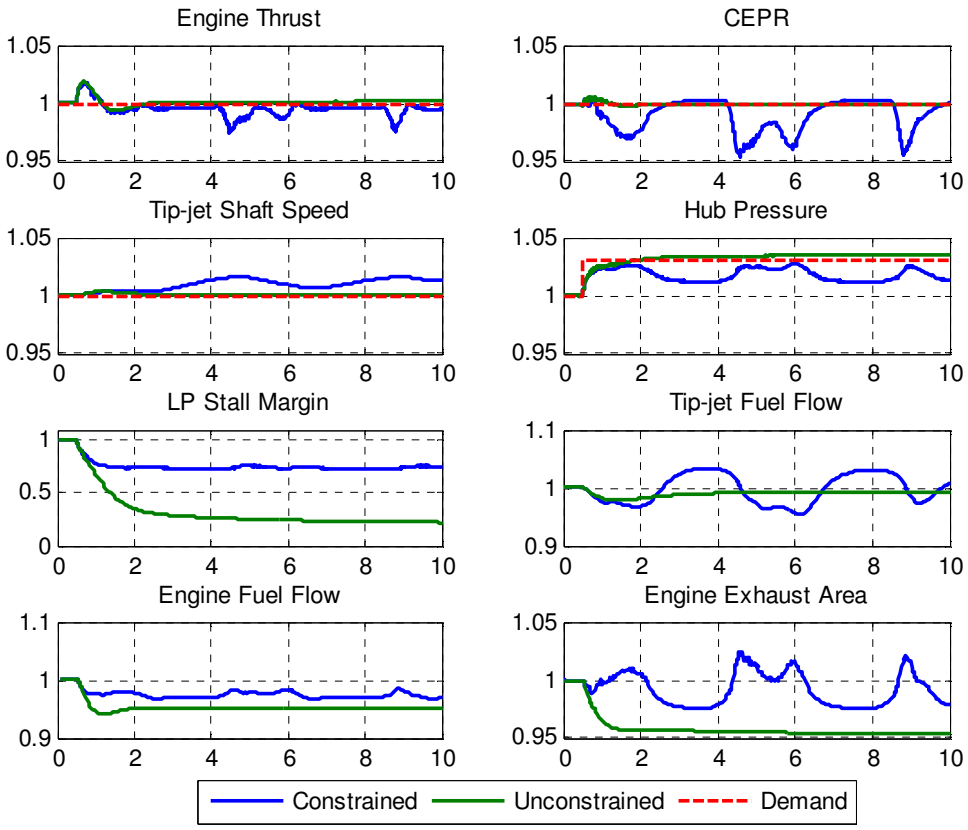

Figure 112: Distributed MPC Control Performance for Stall Margin Limit Avoidance

8.4.2.2 Actuator Rate Limits

Figure 113 and Figure 114 below show two rotor load decrease simulations. The former is unconstrained and the latter has a constraint on actuator rate of both the tip-jet fuel flow and exhaust area. The values of the rate limits for both actuators are the same. The rate limits were chosen such the one of the limits would be active during the transient. For the constrained case, the tip-jet fuel flow actuator is operating at its maximum rate of change from the beginning of the transient till almost 12 seconds

afterwards. Looking at these plots it is obvious that the tip-jet shaft speed responds much slower when the actuator rate limits are active.

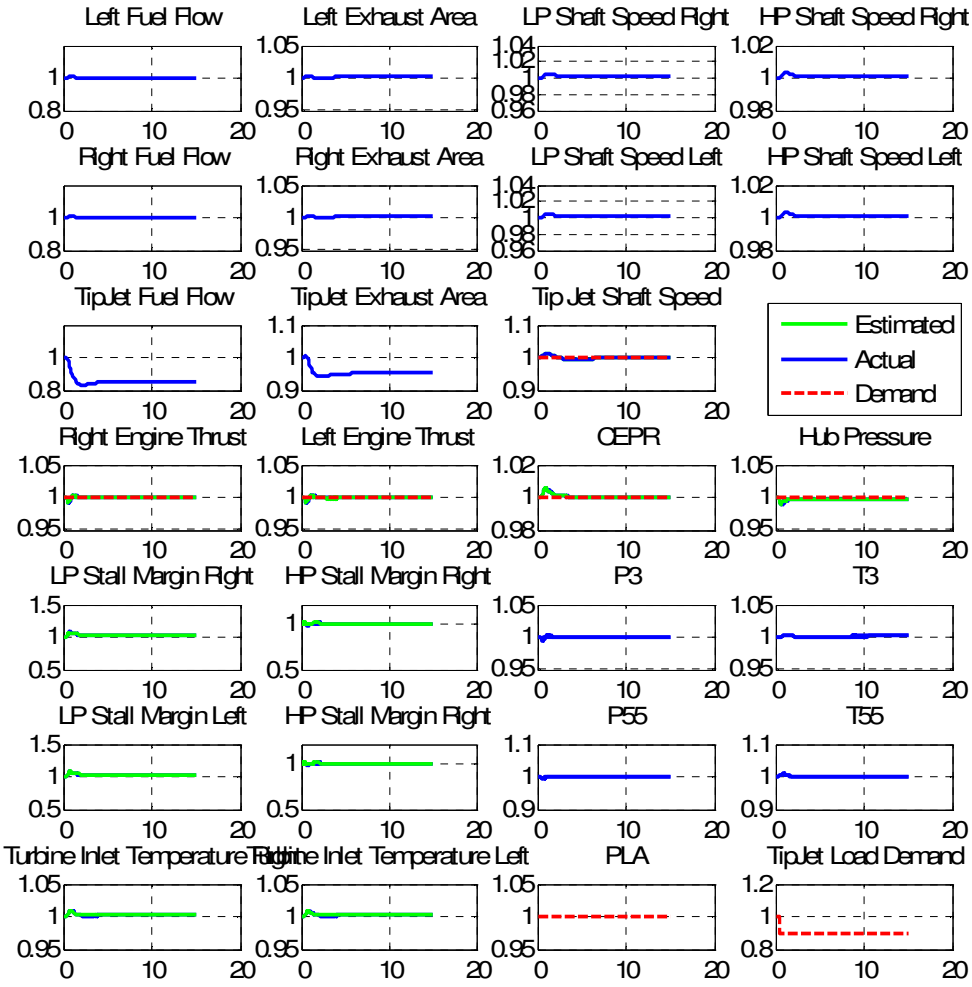

Figure 113: Unconstrained Rotor Load Decrease

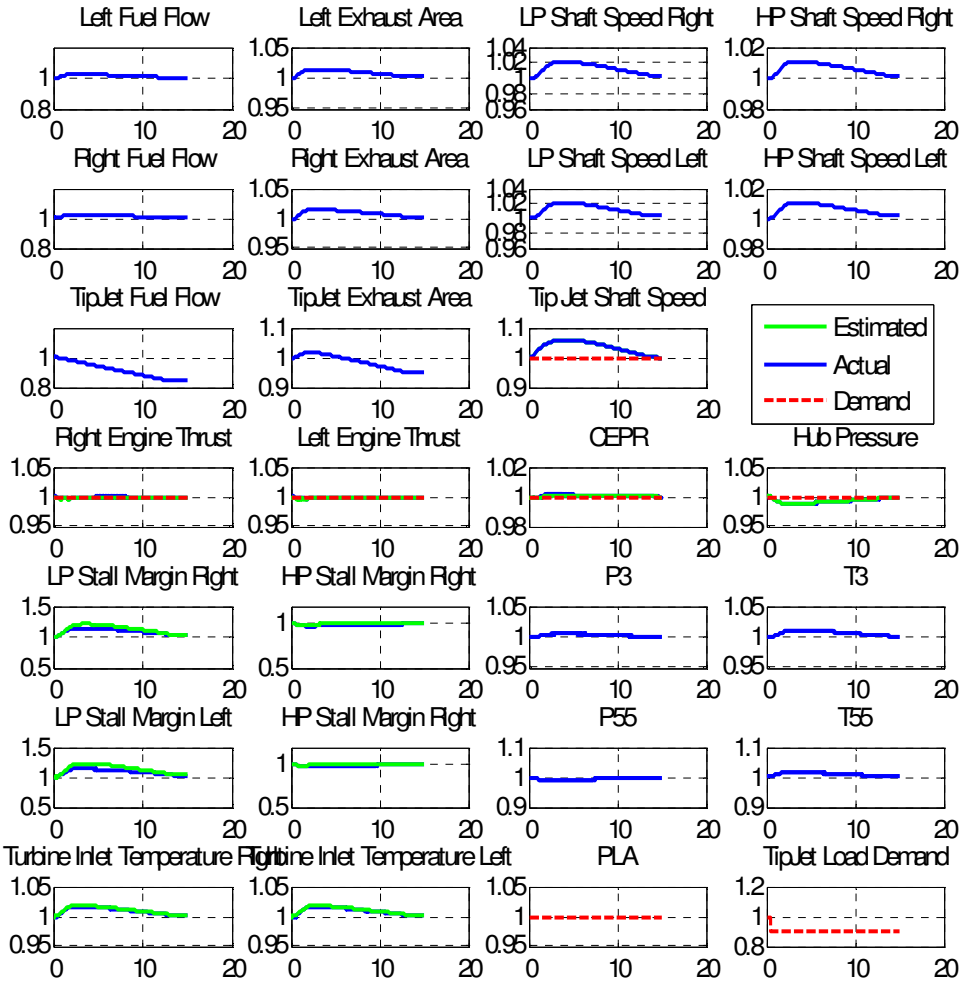

Figure 114: Constrained Rotor Load Increase

Similarly to the stall margin constraint case, directly comparing of the two cases provides some more insight into the differences between the constrained and unconstrained case. Figure 115 below compares some of the control parameters for both the constrained and unconstrained cases. For the constrained case, the tip-jet rotor speed takes almost 6 seconds longer to settle than the unconstrained case with a much more noticeable disturbance in tip-jet shaft speed. This is a direct result of the tip-jet fuel flow being rate constrained. At first glance it appears that tip-jet exhaust area is also

constrained. However the rate of change of the tip-jet exhaust area is slower than the tip-jet fuel flow. Since the fuel flow is larger for the constrained case, the tip-jet exhaust area needs to be more open to maintain target hub pressure. Hub pressure for the constrained case is lower than target. To maintain close to target thrust, the CEPR is correspondingly increased.

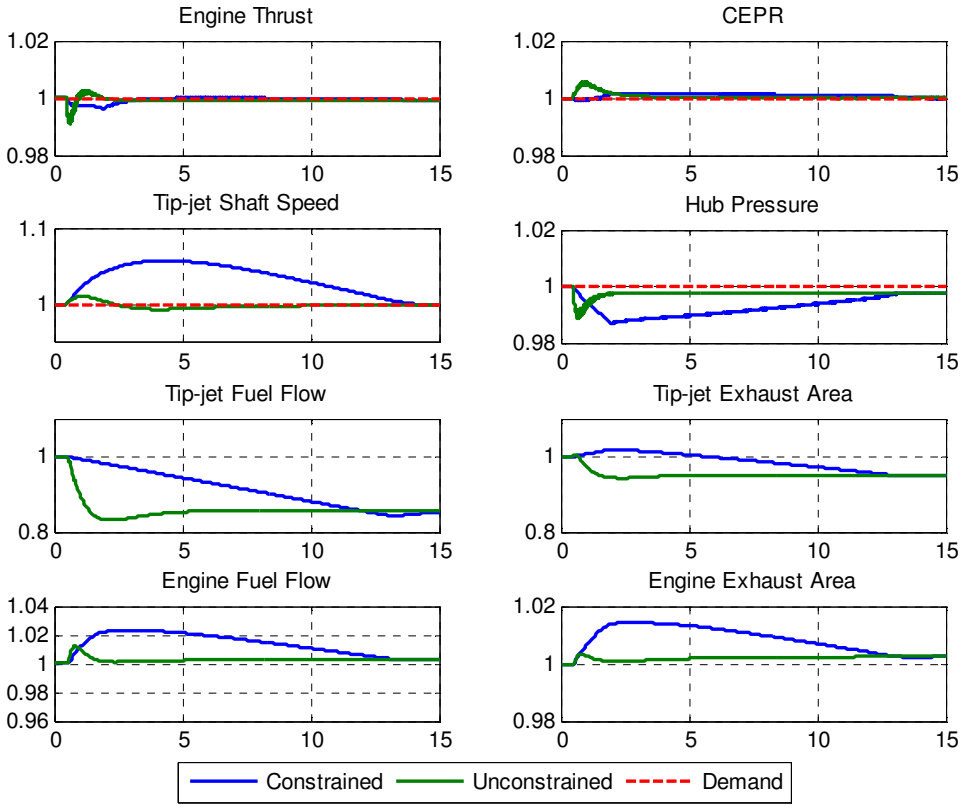

Figure 115: Distributed MPC Control Performance for Actuator Rate Limit Handling

8.5 Distributed MPC Summary

In this chapter, the distributed MPC controller for the tip-jet reaction drive system was developed. Before the controller was sized various design variables were identified. A sensitivity study on the performance of the control to change in the design variables was performed. From this analysis, a final design selection was made. Integrating an optimizer in the design process could result in a final design variable setting that better

matches the desired performance targets. However since significant time is required to generate the control performance for each design variable, care must be taken in this optimization. Given this design, sensitivity to modeling error and disturbance rejection study was performed. After this, different transient simulations were run that captured the response of the system varying throttle settings and changing rotor load. Lastly, simulations were run to understand the constraint handling capability of the controller. Given the definition of all three controllers, a comparison of each can be performed. Conclusions from this comparison will allow the distributed MPC to be integrated into the feasibility study performed in section 2.3.

CHAPTER 9

CONTROL ARCHITECTURE COMPARISON

In this chapter, the performance of the PI controller, centralized MPC, and distributed MPC will be compared. To do this, the comparison is broken up into three general areas. First, both the steady-state and Bode plot analysis from the controller design of the previous chapters will be revisited. After that a comparison of the transient response to throttle and rotor demand changes for all three control architectures will be performed. Lastly, the constraint handling capability of each of the MPC will be compared. Using the conclusions drawn from these comparisons, the distributed MPC can be integrated into feasibility study performed in Chapter 2.

9.1 Controller Design Comparison

. There are two general aspects of performance to which the controllers were designed to. The first aspect is the sensitivity of the steady-state operation of the system to both degradation and modeling errors. The second is the transient response of the controlled system in terms of bandwidth and interactions with changes in demand.

9.1.1 Steady-State Operation

Due to manufacturing variation and aging of the system, the steady-state operation point will vary. The control inputs and outputs were selected to minimize the variation as a result of these two parameters. Additionally, the state estimator of the centralized MPC and distributed MPC was designed to minimize the effects of both modeling error and degradation. It is necessary to first revisit the non-model based versus model performance variation over the life of the system. This analysis will allow comparison of the PI controller with both the centralized and distributed MPC. After this, the sensitivity of the two MPC architectures with respect to modeling errors will be explored.

9.1.1.1 Performance Variation Over Life

Table 33 below compares the performance variance of a non-model based PI controller with the model based centralized and decentralized MPC. In this comparison, the onboard model used in the model based controller is assumed to have no errors. Comparing the mean values of a new and clean system versus a degraded system captures the effects of degradation. The standard deviation in parameters captures the effect of machine to machine variation. A comparison of these two values between model based and non model based control shows the effect of the different types of control.

Table 33: Non-Model Based and Model Based Steady-State Performance Variation

		Non Model Based		Model Based	
Parameter	Age	Mean	Standard Deviation	Mean	Standard Deviation
Engine Thrust	New	0.999	0.008	1.000	0.000
TSFC	New	1.004	0.014	1.004	0.018
LP Shaft Speed	New	1.000	0.000	1.003	0.010
HP Shaft Speed	New	1.000	0.013	1.000	0.010
Turbine Inlet Temperature	New	1.004	0.012	1.004	0.015
LP Stall Margin	New	1.000	0.008	0.997	0.061
HP Stall Margin	New	0.983	0.076	0.987	0.077
Engine Thrust	Degraded	1.008	0.011	1.000	0.000
TSFC	Degraded	1.040	0.014	1.032	0.017
LP Shaft Speed	Degraded	1.000	0.000	0.995	0.008
HP Shaft Speed	Degraded	1.002	0.013	0.999	0.010
Turbine Inlet Temperature	Degraded	1.041	0.011	1.033	0.014
LP Stall Margin	Degraded	1.002	0.007	0.939	0.074
HP Stall Margin	Degraded	1.049	0.075	1.059	0.080

As the system ages, the model based controllers have minimal variation in the expected thrust generated by the engines. For non model based control, as the system ages, the thrust increases by approximately 1%. Additionally there is an approximate 1% variation in thrust resulting from machine-to-machine variation. However, as will be noted below, the incorporation of modeling error affects the variation of thrust over the life of the system. The increase in turbine inlet temperature (and temperatures in general)

over the life with model based control is approximately 1% less than compared with non model based control. Additionally, as the system ages with model based control, the TSFC degrades by about 1% less relative to non model based control.

However, there is the one main drawback to using model based control is the approximate 7% increase fan stall margin variation. This is mainly due to the variations in fan speed while hub pressure remains constant. This factor may result in the choice of a slightly lower design fan pressure ratio, which has an adverse effect on the system performance. However, for MPC minimum stall margin is included as a constraint to ensure safe operation at all times.

9.1.1.2 Sensitivity to Modeling Error

In addition to defining the modeling requirements for an onboard model, the sensitivity to modeling error can be used to supplement the above variation analysis as well as capture any differences between the two MPC architectures. Table 34 below summarizes the sensitivity to modeling error of both MPC architectures. In this table, the sensitivity of each MPC to various types of modeling errors is summarized. The errors simulated were turbine map error, fan map error, reaction drive pressure drop error, and degradation.

Since thrust is being directly controlled by both MPC, the sensitivity of this parameter to modeling error should be minimized. Resulting from degradation, modeling error adds 0.3% variation to thrust over the life of the system that was not accounted for in the above analysis. If there is additional modeling error, especially in the turbine or rotor, the thrust variation for MPC would be much larger than on the PI controller. This fact would require the onboard models of either MPC to have fairly accurate turbine maps and rotor duct loss models otherwise a large portion of the benefit seen from model based control is lost.

Table 34: Model Based Control Sensitivity to Modeling Error

Sensitivity to Modeling Error		Turbine Map Error		Fan Map Error		Reaction Drive Pressure Loss Error		Degraded System	
	MPC Control	Centralized	Distributed	Centralized	Distributed	Centralized	Distributed	Centralized	Distributed
Model-Actual % Difference	Thrust	2.37%	2.23%	0.00%	0.00%	-0.60%	-0.60%	0.27%	0.20%
	LP Stall Margin	7.71%	9.35%	0.20%	0.30%	4.86%	3.90%	2.30%	2.70%
	HP Stall Margin	17.79%	14.45%	0.50%	-0.55%	0.50%	0.60%	5.20%	5.30%
	Turbine Inlet Temperature	-2.60%	-2.50%	-0.10%	0.00%	0.44%	-0.05%	-1.80%	-1.80%
	Hub Pressure	0.24%	0.26%	0.00%	-0.21%	-1.70%	-1.70%	0.92%	0.90%
Target-Actual % Difference	Thrust	2.37%	2.27%	0.00%	0.00%	-0.60%	-0.60%	0.27%	0.20%
	CEPR	0.00%	0.00%	0.00%	0.04%	0.00%	0.01%	0.00%	0.00%
	Ntip	0.00%	0.00%	0.00%	0.00%	0.00%	0.03%	0.11%	0.09%
	Hub Pressure	0.24%	0.26%	0.00%	-0.20%	-1.70%	-1.70%	0.92%	0.90%
Data Match Scalars	Fan Eta	1.002	1.002	0.969	0.967	0.999	0.999	0.980	0.980
	Fan Wc	0.994	0.996	0.971	0.971	1.040	1.042	0.980	0.979
	HPC Eta	0.992	0.993	1.000	1.001	0.999	0.998	0.979	0.979
	HPC Wc	0.989	0.989	1.000	1.001	0.998	0.997	0.978	0.978
	HPT Wc	1.003	1.003	1.000	1.000	1.000	1.000	1.013	1.013

In the state estimator design section the onboard model is not tuned to hub pressure because of the poor matrix properties that would have resulted from including hub pressure. Therefore hub pressure is considered an estimated parameter, like thrust, the estimation of which is sensitive modeling errors. As the system ages, the model overestimates hub pressure by about 1%. The resulting effect of this discrepancy will clearly be seen in the transient simulations performed in section 9.2, where the tip-jet shaft response is much more noticeable on a degraded system.

Over the life of the system, because of degradation modeling error, the LP and HP stall margins vary an additional 2.5% and 5%, respectively. This factor would necessitate the margining of the stall margin constraint in the controllers. In regards to the LP stall margin, using a model based control increases the LP stall margin variation by almost 10% when compared with non-model based control. As with degradation, the stall margin modeling error is significantly increased when the turbine maps are modeled incorrectly. Also, when there is a modeling error in the rotor pressure drop, the model based control will drive the steady-state stall margin to a much lower value. These two

facts also highlight the importance of accurate turbine maps and rotor pressure drop models.

Lastly in comparing the sensitivity to modeling error of the two control architectures, there appears to be a small difference between the two. However when this difference is compared to the sensitivity added because of modeling error, it is minimal. This fact is to be expected because the state estimator (which is used to calibrate the onboard model) used on both controllers is identical. Therefore in terms of sensitivity to modeling error there is no discernable difference between the centralized and distributed MPC.

9.1.2 Bandwidth and Interactions

Table 35 summarizes the bandwidth of all the control parameters for each of the control architectures. Although all the control parameters in the three architectures are not the same, LP shaft speed and thrust control are very similar and can be viewed as analogous. Since an optimizer was used to size the PI controller to match the open loop bandwidth targets, the actual PI controller matches the target performance well. The centralized MPC gets close with to both thrust and CEPR, while hub pressure responds slower and tip-jet shaft speed responds faster. Using a similar optimizer on the centralized MPC design variables should drive the performance closer to the targets. However, since control horizon is proportional to computational burden care must be taken to incorporate that metric into the optimization.

Similar to the centralized MPC, using the results of the sensitivity study a distributed controller that was close the target bandwidth was found. For the final design, both tip-jet shaft speed and CEPR respond faster than desired while hub pressure responds slower. As with the centralized MPC, an optimizer can be used to drive the performance closer to the target values. However, in addition to the capturing the computational burden effect of the control horizon, the CPU requirement for each design

point needs to be accounted for. This necessitates the use of a design of experiment technique to accurately sample the design space while minimizing the CPU cost.

Table 35: Control Architecture Bandwidth Comparison

Controller Bandwidth (rad/s) Comparison	Target	PI Controller	Centralized MPC	Distributed MPC
Fg/N2 Right	3	3.13	2.46	2.9
Fg/N2 Left	3	3.25	2.46	3
Tip-jet Shaft	0.6	0.64	0.8	0.82
CEPR Right	3	2.72	2.78	4.9
CEPR Left	3	2.82	2.78	4.9
Hub Pressure	3	2.86	1.61	2.3

Figure 116, Figure 117, and Figure 118 below are the Bode plot magnitudes for the designs of the PI controller, centralized MPC, and distributed MPC, respectively. These plots can be used to capture both the bandwidth discussed above and the interactions of the control parameters for each of these architectures. Each column in the Bode plot represents the change in demand, whereas each row represents the response of the parameter to a change in demand.

Although none of the interactions present on the PI controller Bode plot in Figure 116 were deemed significant, there were a few that were more significant than others. When engine LP shaft speed demand is changed, the largest interactions are with CEPR of that engine, tip-jet shaft speed, and hub pressure. When hub pressure demand is changed, there are noticeable interactions with both engine shaft speeds and CEPR.

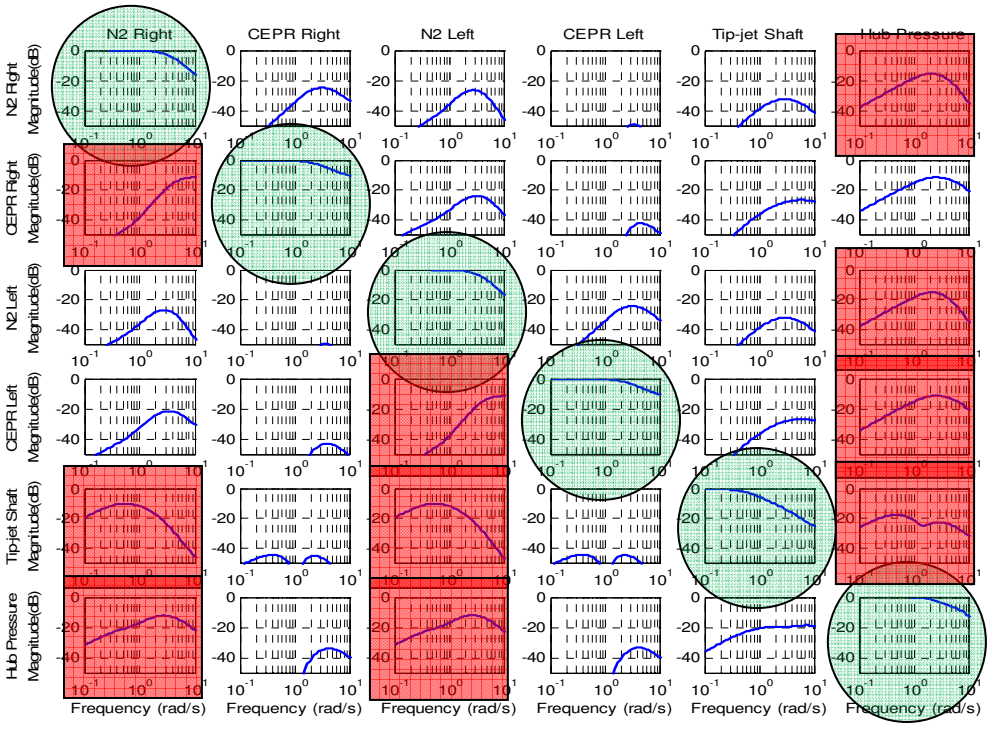

Figure 116: PI Controller Bode Plot Magnitude

Similar to the PI controller, the interactions on the Bode plot of the centralized MPC in Figure 117 did not appear to be any significant interactions; however a few were more noticeable than others. When CEPR demand on a given engine changed, the largest interaction was with the thrust on the engine. Similar to the PI controller with engine shaft speed, hub pressure demand had noticeable interactions with both engine thrusts.

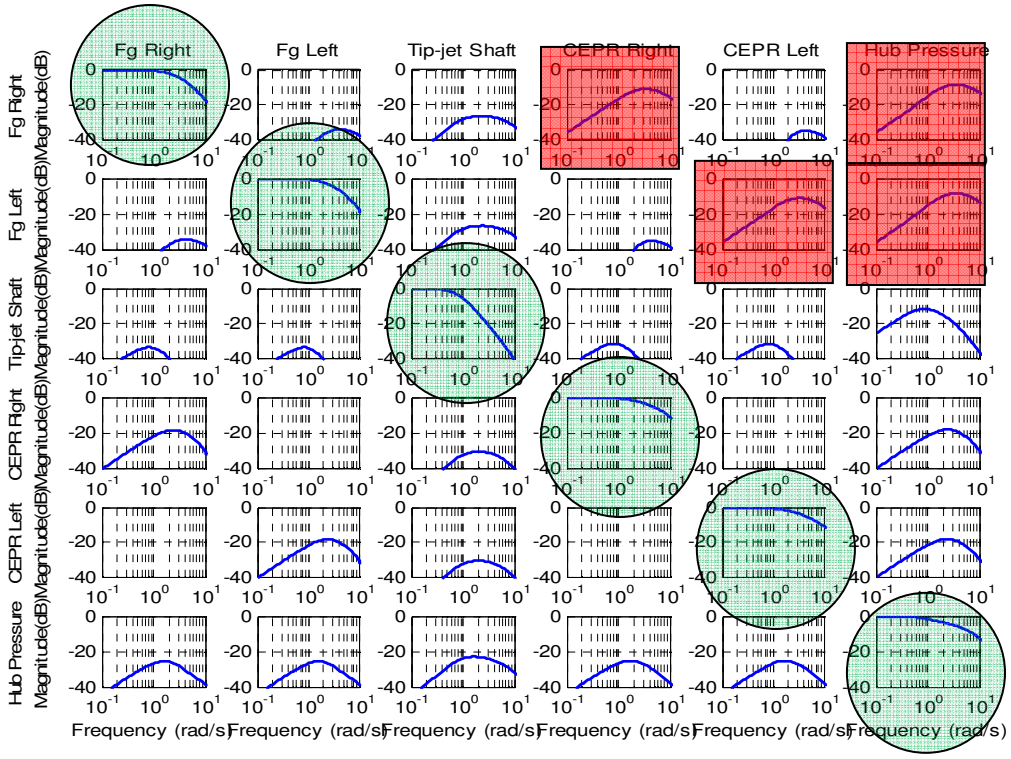

Figure 117: Centralized MPC Bode Plot Magnitude

Minimizing the interactions on the distributed MPC was much more challenging than on the other two control architectures. Since none of the design points explored in the sensitivity study meet all the performance metrics, some extrapolation of trends had to be made. Therefore, these two did not guarantee that the final design met all the performance metrics while minimizing the interactions. Looking at the final design of the distributed MPC shown in Figure 118, there were a handful of noticeable interactions. When engine thrust demand was changed, both the CEPR of that engine and hub pressure had noticeable interactions. And when hub pressure demand was changed, there was a noticeable interaction with tip-jet shaft speed. This interaction was seen during the distributed MPC stall margin constraint handling simulation.

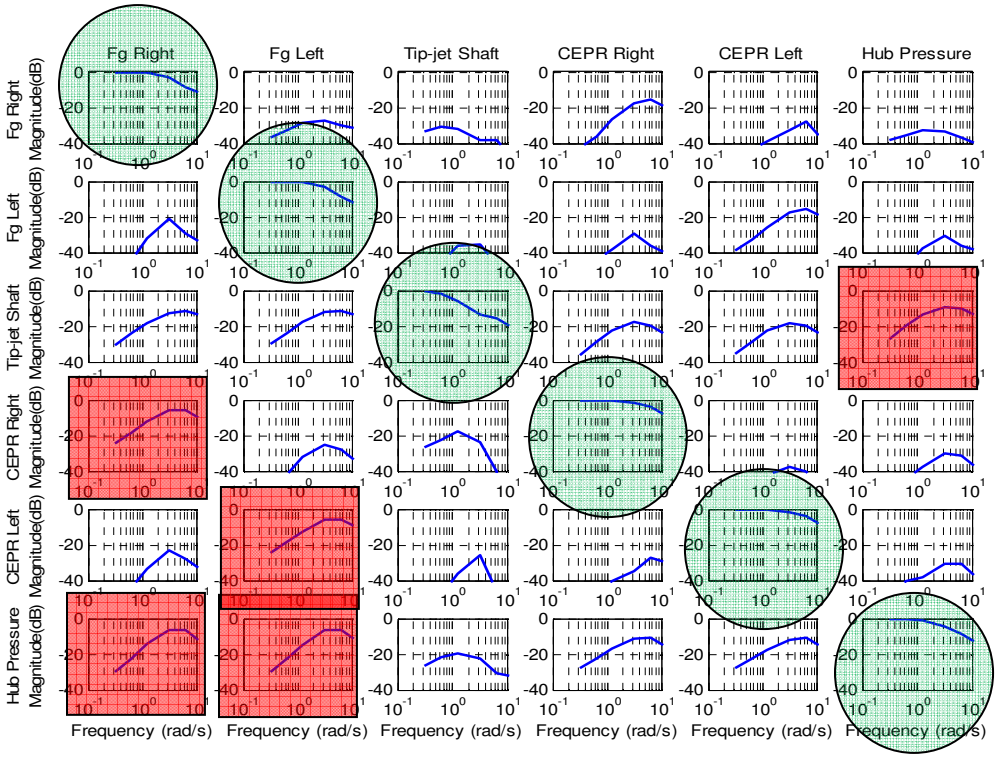

Figure 118: Distributed MPC Bode Plot Magnitude

9.1.3 Performance Summary

From the analysis of both the steady-state performance and Bode plots, the performance of the distributed MPC is very similar to that of the centralized MPC. Integrating an optimization routine into the sizing process of both controllers should lead to performance very close to the desired targets and that of the baseline PI controller. For both MPC it is important that the onboard model turbine map model and the rotor pressure drop model are fairly accurate otherwise additional variation in system performance will be seen.

9.2 Controller Simulation Comparison

The purpose of the simulations in the next few sections is to provide a comparison of the overall response of the each of the three controllers. As was done in the previous chapters the comparisons of the response to varying throttle settings, rotor load changed, and limit avoidance will be explored in detail.

9.2.1 Rotor Load and Throttle Demand Change Simulations

The following sections are broken up into analysis of engine throttle changes, rotor load demand changes, and a combined engine throttle and rotor load demand changes. The engine throttle changes provide an opportunity to analyze how the control responds when multiple demands are changed simultaneously. For a tip-jet reaction drive system, a change in rotor load demand provides a natural opportunity to analyze disturbance rejection. And the third section will combine the two transients. From these comparisons, conclusions can be drawn regarding the response of the different controllers.

9.2.1.1 Engine Throttle Changes

Figure 119 and Figure 120 below contain a comparison of an engine throttle increase for both a new and clean and degraded system on all three control systems. Comparing both the centralized and distributed MPC, the largest difference appears in the hub pressure response. The faster response of the distributed MPC hub pressure is a result of both the higher bandwidth and larger hub pressure interactions with changes in other control parameter demands. Comparing the PI controller with both MPC controllers, the most noticeable difference is the tip-jet shaft speed response. For the new and clean system with a PI controller, the tip-jet shaft speed has about a 1% disturbance while there is minimal disturbance for either of the MPC controllers. This fact can be partially explained by the faster bandwidth of the tip-jet shaft speed controllers. Another reason is that the changes to the engine subsystem are captured in the onboard model of both the centralized and distributed MPC and the control then responds to minimize the interaction of the engine subsystem with the tip-jet subsystem.

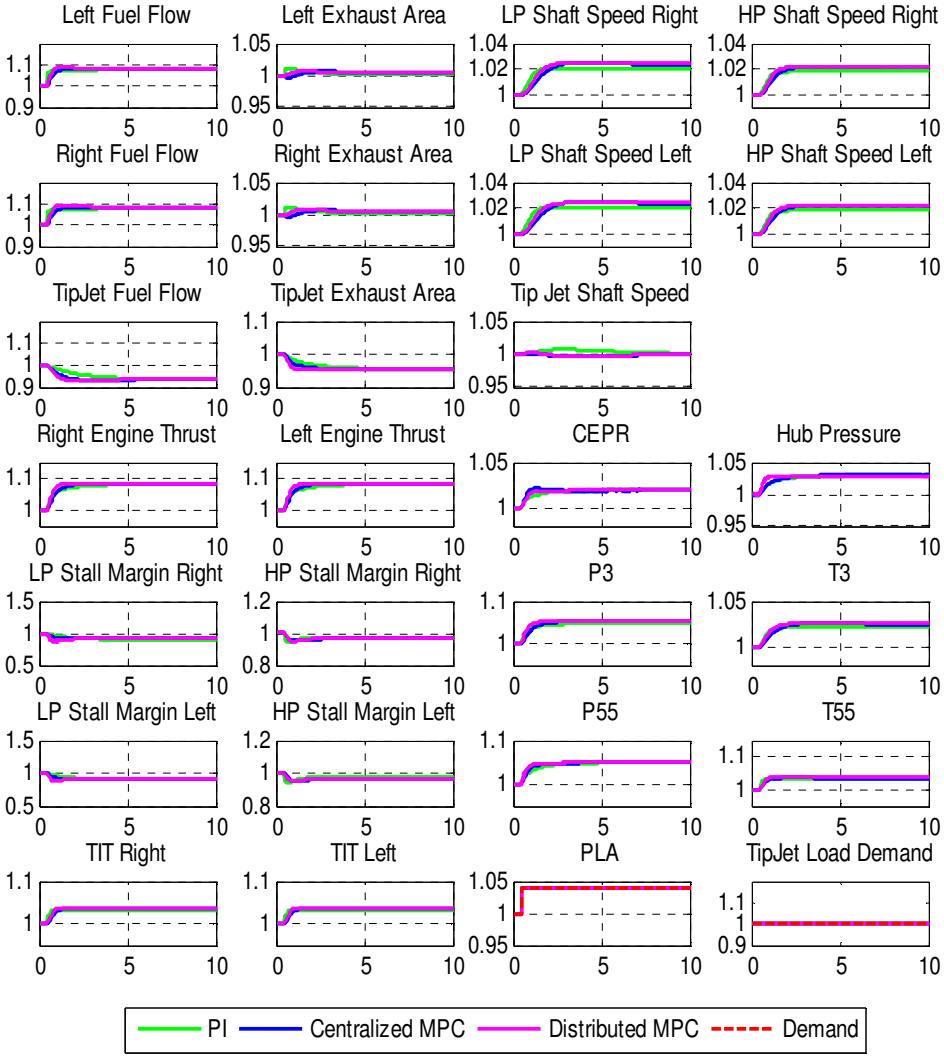

Figure 119: Engine Throttle Increase for New and Clean System

For both the MPC controllers, the disturbance in tip-jet shaft speed was much more noticeable on the degraded system. This is because the degradation modeling error overestimates the hub pressure resulting in the MPC believing there is more torque on the rotor than there really is. Therefore the requested control input change would be different than if there were no modeling error; the resultant tip-jet shaft speed disturbance is much more noticeable. Of interest is the fact that disturbance on the PI controller is in the

opposite direction of that of the MPC. This may be due to the faster tip-jet fuel flow response and the slower LP shaft speed response.

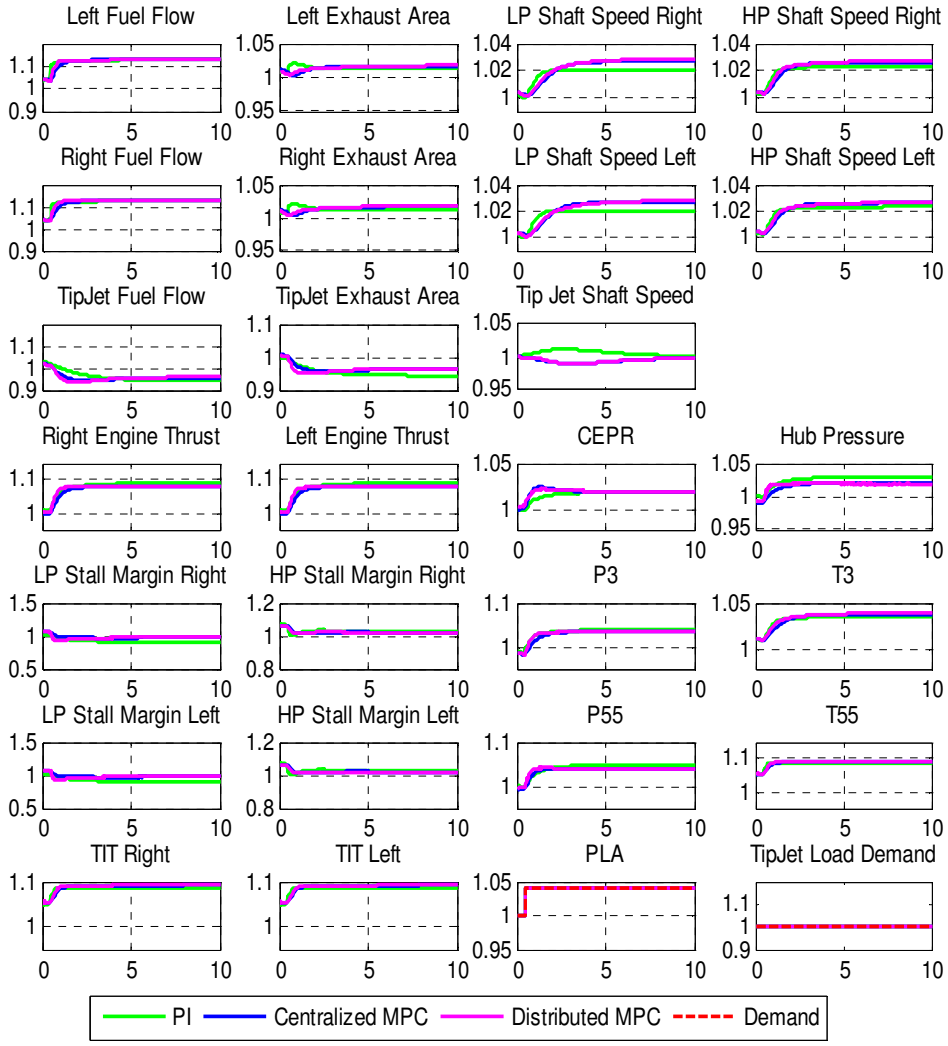

Figure 120: Engine Throttle Increase for a Degraded System

9.2.1.2 Rotor Load Demand Changes

Figure 121 and Figure 122 below contain a comparison of rotor demand increase for both a new and clean and degraded system on all three control systems. Similar to the engine throttle transients, there is minimal difference between the centralized and

distributed MPC. Comparing the PI controller with both MPC controllers, the most noticeable difference is again in the response of the tip-jet shaft speed. For the new and clean system with a PI controller, the tip-jet shaft speed has almost a 5% disturbance in shaft speed with a settling time around 8 seconds. For both MPC, there is a much smaller 1-2% disturbance in shaft speed with a settling time around 4 seconds. As with the engine throttle simulations, this fact can be explained by the faster bandwidth of the tip-jet shaft speed controllers.

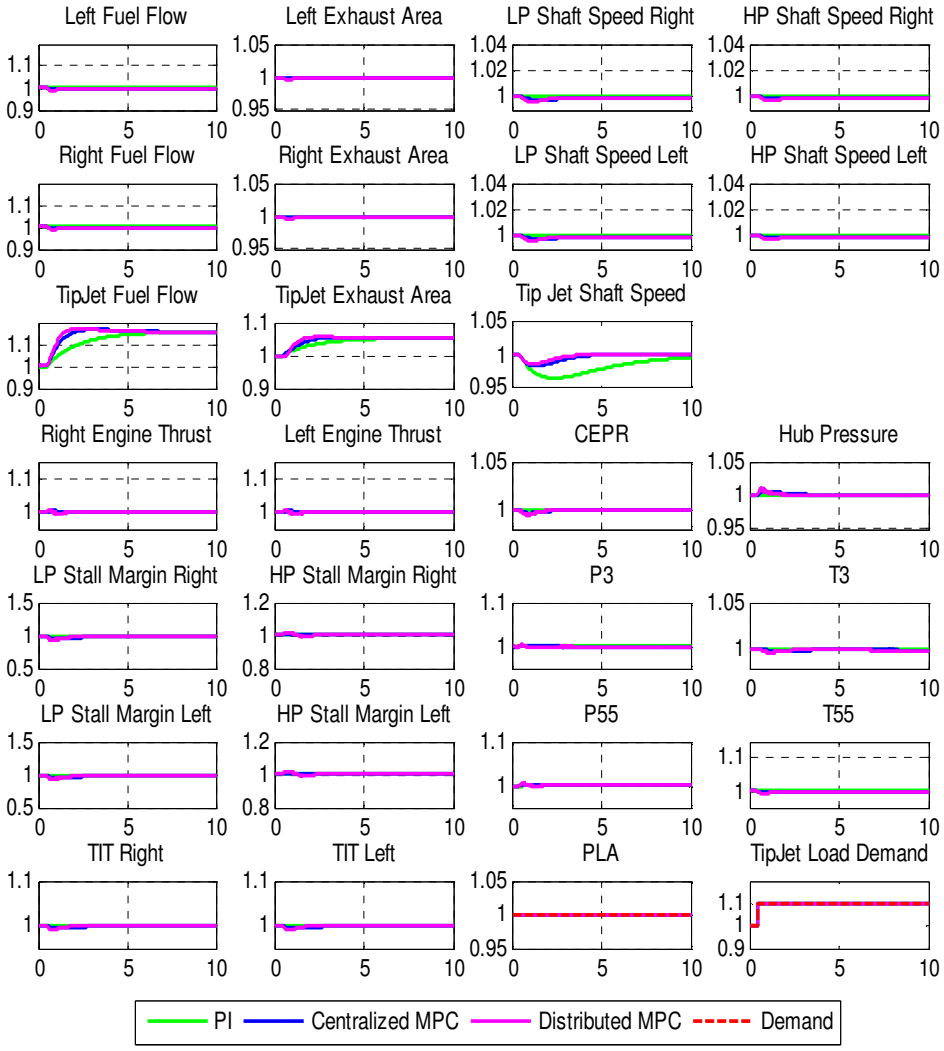

Figure 121: Rotor Load Increase for a New and Clean System

For the degraded case there did not appear to be a significant change in the response of the PI controller; however, for both MPC there is a larger disturbance in tip-jet shaft speed with a settling time of around 5 seconds. This is a result of the degradation modeling error where the actual pressure ratio is slightly lower than estimated by the model resulting in a lower estimated torque in the rotor.

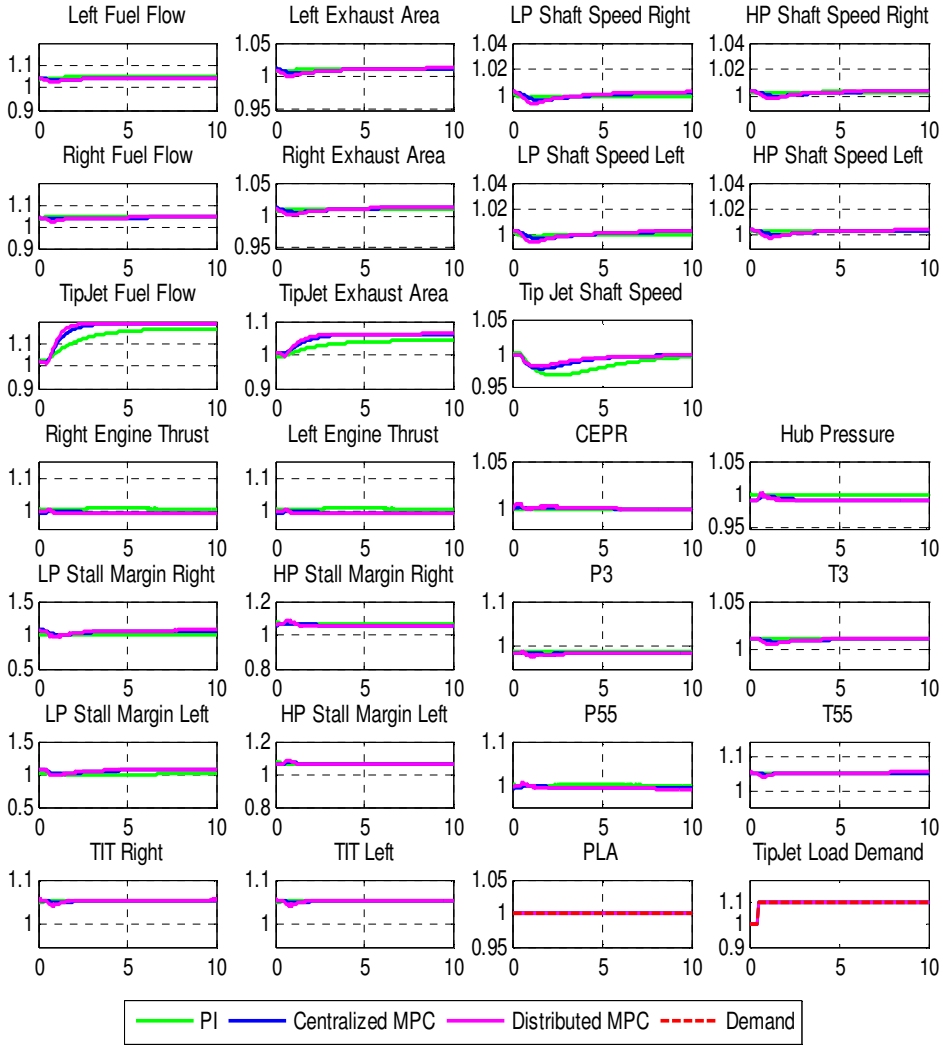

Figure 122: Rotor Load Increase for a Degraded System

9.2.1.3 Engine Throttle and Rotor Load Demand Changes

Figure 123 and Figure 124 show the combined engine throttle and rotor load demand simulations for all three control architectures. As has been observed in the previously chapters, there do not appear to any significant additional interactions for any of the controllers as a result in combining the throttle change and rotor load. And all the trends noted in the previous section hold true for these simulations.

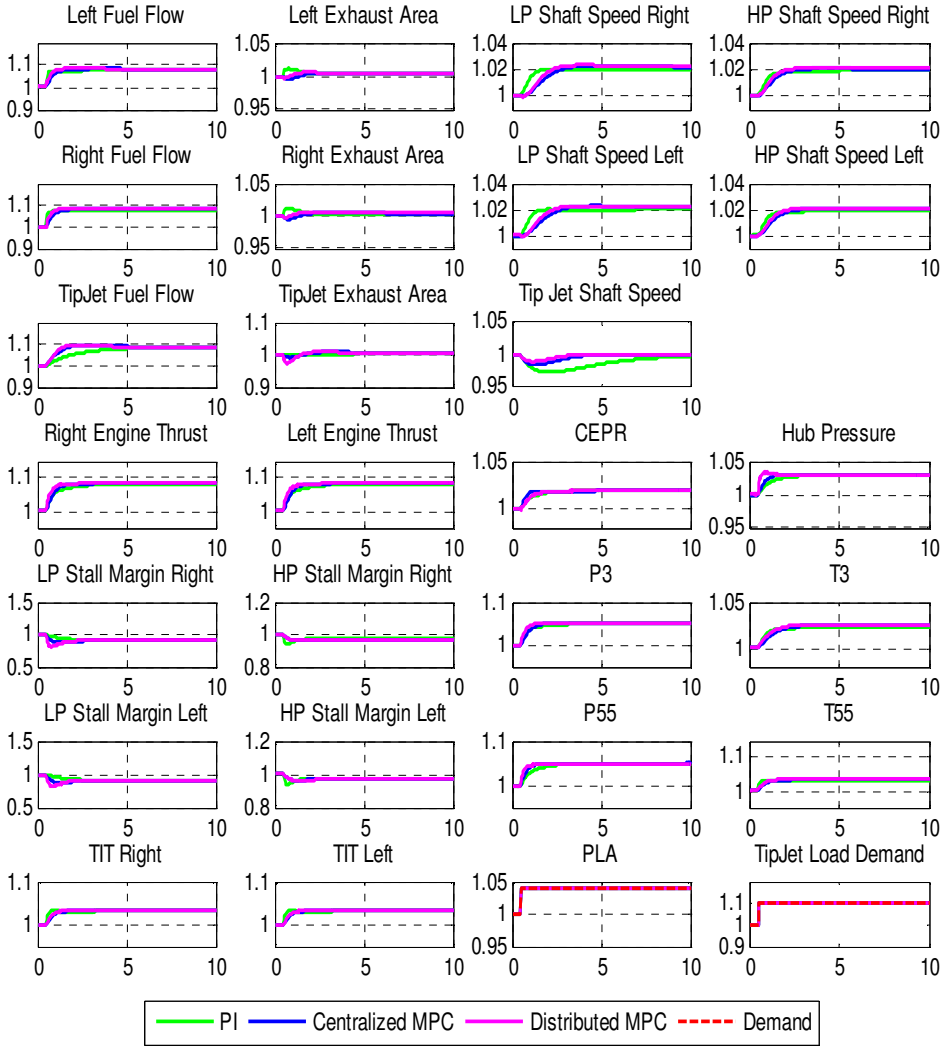

Figure 123: Engine Throttle and Rotor Load Increase for a New and Clean System

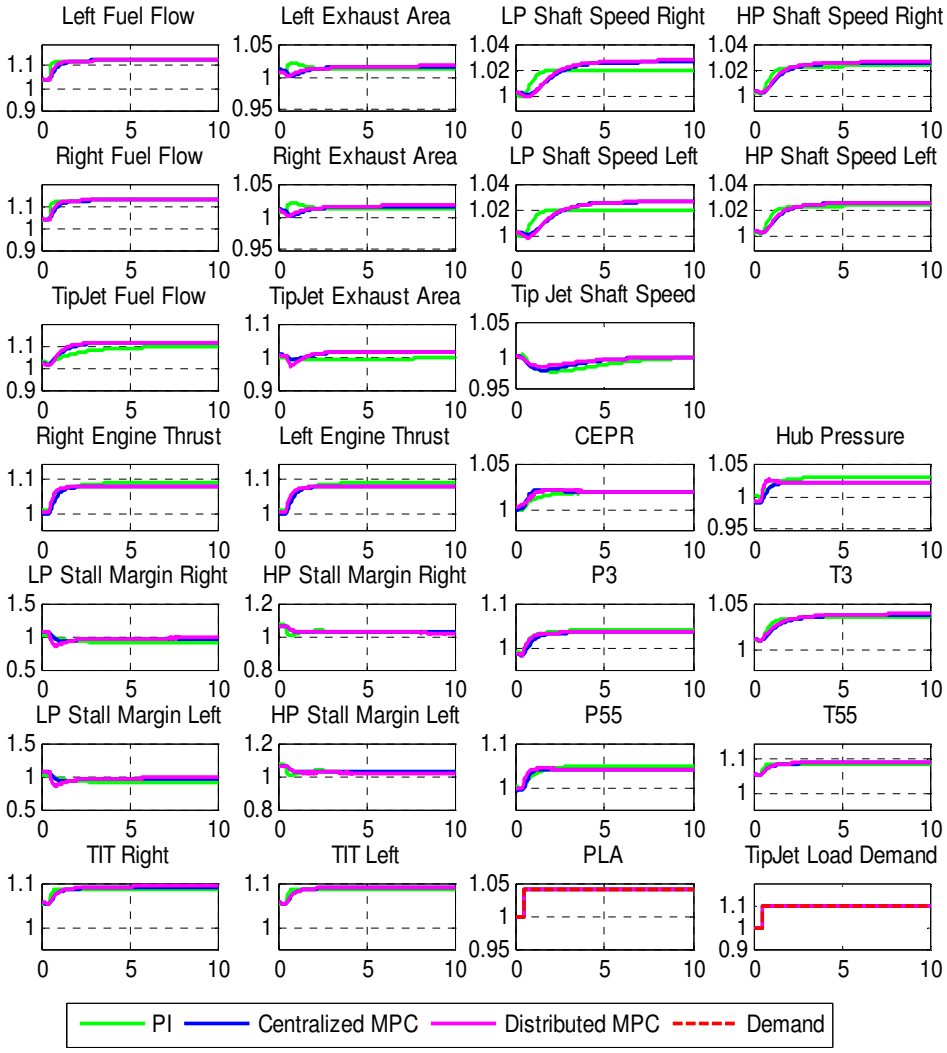

Figure 124: Engine Throttle and Rotor Load Increase for a Degraded System

9.2.1.4 Demand Change Summary

From analysis of the demand change simulations above, it is evident that there is minimal difference between the response of the distributed MPC and centralized MPC. The main difference is seen in the hub pressure response; however that is largely due to the final settings of the design variables. Integrating an optimization into the sizing process should minimize this difference. As the system degraded, the tip-jet shaft speed

response of each of the MPC controllers was slower. This was a result of the degradation modeling error of the hub pressure.

9.2.2 Constraint Handling

As has been stated throughout the dissertation, a primary driver for the use of MPC is the direct inclusion of constraints in the control algorithm. In both the centralized and distributed MPC chapters, simulations were run which showed how the inclusion of constraints affected the response of the controllers. In this section, the constrained response of each of the MPC will be compared.

9.2.2.1 Stall Margin Handling

Figure 125 below shows the comparison of stall margin constraint handling for all three control architectures. Although a hub pressure demand was used to simulate stall margin handling, the non-model based versus model based nature of the controllers required different magnitude hub pressure demand changes to capture the constraint handling capability fully. This can be seen by fact that the constrained hub pressure for the PI controller is much larger than that of both the MPC.

It can clearly be seen that the distributed MPC response to a constraint is very unsteady. This fact is primarily due to the interaction between tip-jet shaft speed and hub pressure. Fine tuning the controller design variables should be able to reduce the interactions. Additionally adding LP stall margin to the tip-jet fuel flow optimization delayed the onset of the oscillations. Lastly, during some very preliminary studies of the distributed MPC constraint handling, these oscillations were not seen. However, the design configuration used for those cases did not result in favorable control performance metrics. Although the responses of the distributed MPC have significant oscillations, the stall margin constraint is never violated. And it is believed that fine tuning the controller would lead to better stall margin constraint handling.

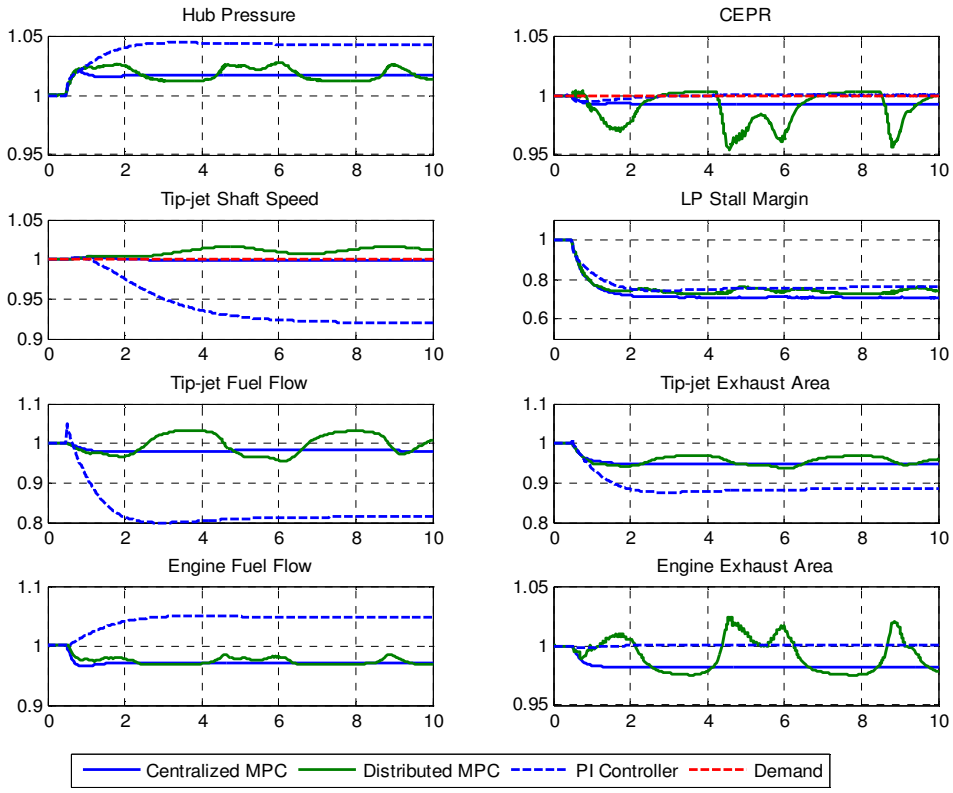

Figure 125: MPC Stall Margin Constraint Handling Comparison

The aspect of the PI controller response to a stall margin constraint that stood out the most was the tip-jet shaft speed. When the stall margin constraint was encountered the eventual steady-state tip-jet shaft speed was almost 10% below the target value. This compares to a 0% deviation and 2% deviation from the target for the centralized and distributed MPC. The further the tip-jet shaft speed deviates from the target, the more the aerodynamic properties of the rotor blades may change which may lead to some negative aspects to the vehicle handling properties. As opposed to the distributed MPC where changes in design variables can drive the tip-jet shaft response closer to target, the PI controller response to the constraint is fixed by the separate controller requirement of the control architecture and cannot be changed. This example conclusively demonstrates a significant reason why a MPC would be chosen in favor of a PI controller.

9.2.2.2 Actuator Rate Limit Handling

Figure 126 shows the response of the two MPC to encountering a tip-jet fuel flow actuator rate limit. When the constraint is encountered on the distributed control the response of the system is much slower. This is because each actuator is updated every other time-step and therefore would need twice as long to reach the final steady-state point. This difference could be alleviated by adjusting the distributed MPC control optimization. The current algorithm forces the control input to move the full demanded value during the time-step of the optimization and then held constant during the time-step where it is not being optimized. However, if the change in control move is integrated over two time-steps with half the desired change implemented at each time-step the response of the distributed controller should approach that of the centralized MPC

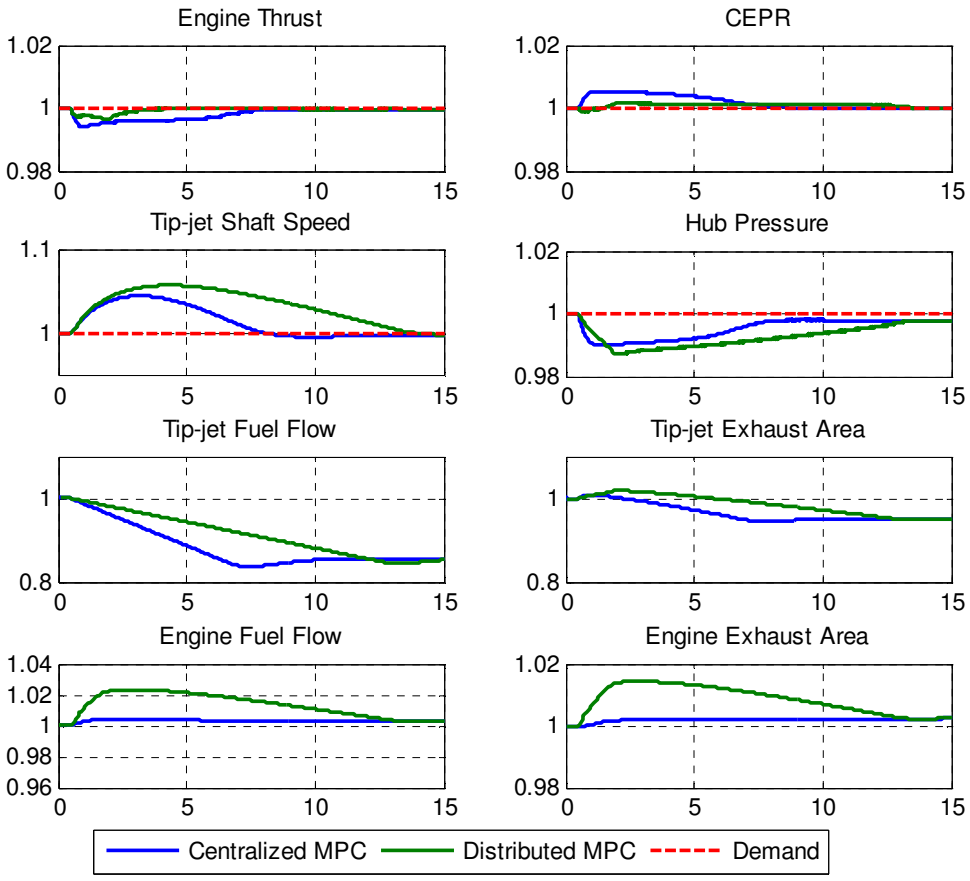

Figure 126: MPC Actuator Rate Limit Constraint Comparison

9.2.2.3 Constraint Handling Summary

From the above analysis it is seen that the distributed MPC can directly handle constraints. However in handling those constraints, there is some undesirable response relative to the centralized MPC. Fine tuning the design variables or slightly altering the algorithm should improve the constraint handling capability of the distributed controller and drive it toward the performance of the centralized MPC.

9.3 **Multivariable Control Feasibility Study**

The conclusions drawn in the previous section can be used to integrate the distributed MPC into feasibility study performed in section 2.3. Table 36 below shows the feasibility study with the distributed MPC included. As was noted, there was minimal difference in the steady-state operation and transient of the distributed and centralized MPC. Therefore the performance and control metrics ranking of the distributed MPC should be the same as that of the centralized MPC. In the derivation of the distributed MPC algorithm in chapter 5, the computational burden of the centralized MPC can be reduced by almost an order of magnitude. Therefore the red seen in the centralized MPC ranking in computational burden can be improved to a yellow/red. Although the distributed MPC handled constraints directly, the response relative to the centralized MPC was worse. Improvements to the sizing process and slight changes to the MPC algorithm should improve the constraint handling. Therefore the distributed MPC ranking in regards to constraint handling is slightly lower than that of the centralized MPC. In regards to verification and validation and costs, the distributed MPC should perform similar to the centralized MPC. From these results, it can be concluded that the distributed MPC performs similar to a centralized MPC with a significantly reduced computational burden.

Table 36: Updated Tip-Jet Reaction Drive Control Feasibility Metrics

Tip-Jet Reaction Drive Control Feasibility Study	Non-Model Based PI	Model Based PI	Non-Model Based H-Infinity	Model Based H-infinity	Linear Quadratic Gaussian	Model Predictive Control	Distributed Model Predictive Control
Direct Parameter Control							
Optimal Control							
Performance							
Robustness							
Constraint Handling							
Gain Scheduling							
Computational Burden							
Cost							
Verification and Validation							
Maintenance Costs							

CHAPTER 10

SUMMARY, CONCLUSIONS AND FUTURE WORK

10.1 **Summary**

The research in this dissertation was motivated by the need to reduce the computational burden of a centralized MPC for a tip-jet reaction drive system. The faster dynamics of this system require an extremely short sampling rate (on the order of 20ms), and the slower dynamics require a longer prediction horizon. This, coupled with the fact that the tip-jet reaction drive system has multiple control inputs, makes the integration of an online MPC algorithm for the tip-jet reaction drive challenging. It is this set of circumstances that acted as motivation for the formulation of the first research question. ***How can the dimensions of the MPC problem for the tip-jet reaction drive system be reduced to minimize the computational burden, thus making MPC a more feasible alternative?*** To answer this question, a multiplexed scheme was proposed. However, limitations in this scheme led to the formulation of the second research question. ***For the tip-jet reaction drive system with highly interactive subsystems and a large number of multiplexed control inputs, how can the MPC problem be broken down or divided to minimize the control performance loss due to multiplexing?*** In answering this question it was hypothesized that combining both a multiplexing and feasible cooperative control into a single algorithm would reduce the computational burden of the MPC problem for the tip-jet reaction drive system while maintaining similar control performance to a centralized MPC algorithm.

The detailed analysis of the matrices of both the centralized and proposed MPC algorithms in Chapter 5 were used to confirm the computational burden reduction of the MPC problem of the tip-jet reaction drive system. While no constraints are present, the computational burden is reduced by a factor of 6 when the proposed MPC is used. Since

only one control is being optimized at a time on the proposed MPC, when control input constraints are included the computational burden reduction approaches a factor of 10.

The control architecture comparison in Chapter 9 confirmed that the performance of the proposed MPC was very similar to that of a centralized MPC or PI controller. In terms of bandwidth, disturbance rejection, and sensitivity to modeling error, there is not a noticeable different between the two MPC controllers. Although the constraints are handled adequately for the proposed controller, improvements can be made in the design and sizing process to drive the constraint handling towards that of the centralized MPC. Given these observations, the hypothesis of the dissertation, that the proposed MPC reduces computational burden while maintaining close to centralized MPC performance, has been confirmed.

10.2 **Conclusions**

The previous section summarized how the motivation led to the formulation of the hypothesis on which this dissertation is based. In the process of confirming the hypothesis, the primary capabilities and limitations of the feasible cooperative multiplexed MPC were identified. Both of these will be highlighted in the next section. In addition to the capabilities and limitations, feasible cooperative multiplexed MPC makes contributions to the state of the art of MPC. These contributions will also be covered in this section.

10.2.1 Capabilities

The primary capability of the feasible cooperative multiplexed MPC is reducing the computational burden of the MPC problem for the tip-jet reaction drive system by approximately a factor of 5-10 while maintaining similar control performance to a centralized MPC algorithm. Although this dissertation focused on a tip-jet reaction drive system, this capability can be more generally extended to systems with a large range of

dynamics, a large number of control inputs, and which can be broken down into subsystems. For this class of systems, this control provides good control performance on par with a centralized MPC while reducing the computational burden.

In addition to performing similarly to a centralized MPC, this controller offers some benefits when compared with a non-model based PI controller. The use of an onboard model allowed reduced variance in thrust over the life of the system as well as operating at lower TSFC and system temperatures as the system degrades. The nature of MPC allows for direct multivariable constraint handling in a single controller. And lastly the optimal nature of the controller offers consistent control response in the presence of changes in system dynamics or operating point.

10.2.2 Limitations

One primary limitation of the feasible cooperative multiplexed MPC which was highlighted in this dissertation is the time required to perform the control design. It took up to 12 hours to generate a Bode plot for a single design point. Since the feasible cooperative multiplexed MPC is time variant, the control could not be linearized. This required the Bode plots used in the control performance analysis to be generated by hand. For every second of a simulation it required anywhere from 100-150 seconds of clock time. For each output parameter demand change, multiple period sine waves of varying frequency from 10 rad/s to 0.1 rad/s were input into the controller to obtain the output parameter response to changes in the demand. The lower frequency sine waves were the most time consuming to generate, requiring almost 50 minutes to simulate. There were three general factors that contributed greatly to the large time requirement: the building of the MPC matrices, the Simulink environment, and computer processor speed.

Since a different control input is being optimized at each time-step, a different set of MPC matrices is required. Currently, the MPC matrices used in the optimization are rebuilt from scratch at each time-step. Toggling between prebuilt MPC models which

represent the MPC matrices for a specific control input may reduce the time-requirement for building the matrices. The Simulink environment used to simulate the MPC also had an effect on the time required to simulate a design point. The feasible cooperative multiplexed MPC was represented in Simulink using the Matlab mpcmove function placed in a custom S-function. The mpcmove function is a generic function used to simulate a single time-step of a MPC algorithm. Because of its generic nature this function contains some functionality (and added CPU cost) that may not be needed for the feasible cooperative multiplexed MPC for a tip-jet reaction drive system. Efficiently building the code required to simulate a time-step to represent the calculations required specifically for a tip-jet reaction drive system should improve the time required to simulate a time-step. The design points were run on multiple computers whose processors were either a 2.10 GHz Intel Core2 Duo with 4 GB of RAM or a 3.00 GHz Pentium 4 with 504 MB of RAM. The use of the computer with the Intel processor reduced the time required to run a design point to around 8 hours as compared to 12 hours for the computer with the Pentium processor.

Another limitation of one of the feasible cooperative multiplexed MPC is the ability of the controller to handle constraints. As demonstrated in the previous chapters, the feasible cooperative multiplexed MPC was able to avoid exceeding a constraint but the controller did not perform optimally with either oscillations (stall margin handling) or sluggish performance (actuator rate limit handling) noted on many parameters. The oscillation issue may be overcome by integrating constraint handling into the control design process. The sluggish performance issue may be overcome by changes in the MPC algorithm.

Another limitation of the feasible cooperative multiplexed MPC is its ability to reduce the computational burden on systems with very large interactions. One of the requirements of the feasible cooperative multiplexed MPC is to break down the system into multiple subsystems. To ensure optimal control performance in the presence of

interactions between these subsystems, the feasible cooperative multiplexed MPC iterates potentially multiple times. If the interactions are so large that greater than 5 or 10 iterations are required, the computational burden benefit of this algorithm is lost.

Another potential limitation to any distributed MPC is the sensitivity to information delay. All the experiments performed in this dissertation assumed to the information delay between the distributed MPC was one control sampling interval. However factors such as physical distance, wiring technology, or sampling intervals may introduce a larger information delay. As the information delay becomes large relative to the control sampling interval, there may be a large increase in the interactions in the response as seen when the other subsystem control inputs were assumed constant. Therefore for a given system, a sensitivity study with respect to information delay may be required to determine the appropriate hardware needed to minimize this effect.

10.2.3 Contribution to the State of the Art

The primary contribution to the state of the art in this dissertation is the development of the feasible cooperative multiplexed MPC which reduces the MPC computational burden for a tip-jet reaction drive system by almost an order of magnitude while maintaining control performance similar to that of a centralized MPC algorithm. A secondary contribution is the creation of a Matlab and Simulink based framework that can be used for the control performance design and analysis as well as the comparison of multiple control architectures.

10.3 Future Work

There are several areas for future work. The most obvious area is the integration of an optimizer in the sizing and design of the MPC. This would drive the performance of the MPC much closer to the performance targets while taking much of the guesswork out of the design process. In addition to control response, computational burden and

constraint handling should be included in the optimizer's objective function. This will result in a controller that meets all the multivariable design concerns, while ensuring the computational burden is at a minimum.

The proposed MPC was designed and analyzed at a single operational point and used a simple linear onboard model. The design and analysis can be extended to different operating points across the flight envelope. Additionally, the feasibility of integrating different types of onboard models can be explored.

The model in the MPC algorithm was assumed to be time invariant. However, when the tip-jet reaction drive system changes mode, this assumption may not be valid. Therefore addressing situations where the model is time variant may be an area of future work.

APPENDIX A

TIP-JET REACTION DRIVE SYSTEM NPSS .MDL SOURCE CODE

```
//
//-------------------------------------------------------------------------
//                                                          |
//   File Name:     TIPJET.mdl                                    |
//   Date(s):       September 21, 2008                              |
//   Author:        Brian Kestner                                 |
//                                                          |
//   Description:   Tip-Jet Reaction Drive Model              |
//                                                          |
//-------------------------------------------------------------------------


setThermoPackage("GasTbl");

//-------------------------------------------------------------------------
//                        User-Defined Elements
//-------------------------------------------------------------------------
#include <src\Emission.int>;
#include <src\CoolIt.bckup>;
#include <src\Supersonic_Inlet.int>;
#include <src\Supersonic_Inlet_sub.int>;
#include <src\VolumeDynamics.int>;


//-------------------------------------------------------------------------
//                         Output Data Viewers
//-------------------------------------------------------------------------
#include <view\SBJ.view_page>
#include <view\SBJ.view_ei>
#include <view\SBJ.view_flops>
#include <view\russ_page_160.view>
#include <view\sbj_160_row.view>
#include <view\sbj_160_guess.view>

//-------------------------------------------------------------------------
//                         Shaft Element Extension
//-------------------------------------------------------------------------

class Shaft1 extends Shaft{
   real Ndes=0;
   real NqNdes=0;
   real NqNdesPct=0;
```

```
  //variableChanged logic
  void variableChanged( string name, any oldVal ) {
    Shaft::variableChanged( name, oldVal );
    if (name=="switchDes"){
      if (switchDes=="DESIGN"){
        Ndes=Nmech;
        NqNdes=1.0;
        NqNdesPct=NqNdes*100;
      }
    }
  }
  //Engineering Calculations
  void calculate() {
    Shaft::calculate();
    NqNdes=Nmech/Ndes;
    NqNdesPct=NqNdes*100;
  }
}//ends Shaft1
//-------------------------------------------------------------------------
//                      Model Definition
//-------------------------------------------------------------------------
MODELNAME = "Tip Jet Reaction Drive System";
AUTHOR = "Brian Kestner";

//-------------------------------------------------------------------------
//                      Left Engine
//-------------------------------------------------------------------------

Element FlightConditions AmbientLeft {
  alt = 0;
  MN = 0.00;
  W = 209.0;
  W = 80.0;
  void preexecute() {

                if(VolumeLeft1.switchDyn == "ON" && switchDes == OFFDESIGN) {
                //cout<<"VolumeLeft13.dWqdt = "<<VolumeLeft13.dWqdt<<endl;
                //cout<<"VolumeLeft16.dWqdt = "<<VolumeLeft16.dWqdt<<endl;
                                if(time> 1) {
                                //solver.maxIterations = 2;
                //cout<<"time = "<<time<<endl;
                //cout<<"VolumeLeft13.Fl_Itemp.W =
"<<VolumeLeft13.Fl_Itemp.W<<endl;
                //cout<<"VolumeLeft16.Fl_Itemp.W =
"<<VolumeLeft16.Fl_Itemp.W<<endl;
```

```
                //cout<<"VolumeLeft16.dWqdt = "<<VolumeLeft16.dWqdt<<endl;

                }
                        W = VolumeLeft1.Fl_Itemp.W;
                        //cout<<"W = "<<W<<endl;

            }

          }
}


Element Inlet InletLeft {


  real zepr;
  real WC_Schedule, WC_DESIGN;
  WC_DESIGN = 400.00;
  WC_DESIGN = 80.00;
//   W = 380.00;

}

Element VolumeDynamics VolumeLeft1{
ref1 = "InletLeft.Fl_I";
ref2 = "Fl_O";
length = 10;
volume = 100*length;
void calculate () {
   flowSave = VolumeLeft2.Fl_Itemp.W;
         dp = InletLeft.Fl_O.Pt - InletLeft.Fl_I.Pt;
         VolumeDynamics::calculate();

         }

}


Element Compressor FanLeft {


  #include "src\D60fan.ncp";
  real stallSignal = 0;
  S_map.effDes = 0.8445;
  S_map.PRdes = 2.303;
  S_map.RlineMap = 2.10;
```

```
  S_map.NcDes = 1.0;    // set to 1.0 in ncp file R. Denney 6-21-05

}

Element VolumeDynamics VolumeLeft2{
ref1 = "FanLeft.Fl_I";
ref2 = "Fl_O";
//LastVolume = "TRUE";
length = 10;
volume = 100*length;

void calculate () {
   flowSave = VolumeLeft3.Fl_Itemp.W + VolumeLeft13.Fl_Itemp.W;
         //pwr = FanLeft.pwr/C_BTU_PER_SECtoHP;
         pwr = (FanLeft.WcCalc/FanLeft.Fl_I.Wc)*FanLeft.pwr/C_BTU_PER_SECtoHP;
         dp = FanLeft.Fl_O.Pt - FanLeft.Fl_I.Pt;
         VolumeDynamics::calculate();
         }

}

Element Splitter SplitterLeft {
  BPR =1.0;
  BPR =4.0;
  real coreFlow, bypassFlow;
  void preexecute() {

                  if(VolumeRight13.switchDyn == "ON" && switchDes == OFFDESIGN)
{
                        BPR = VolumeLeft13.Fl_Itemp.W/VolumeLeft3.Fl_Itemp.W;
//                      BPR = VolumeLeft13.Fl_Itemp.W/coreFlow;
            }

   }


   void postexecute() {

                        //cout<<"Fl_I.W = "<<Fl_I.W<<endl;
                  if(VolumeRight13.switchDyn == "OFF" ) {
                        coreFlow = Fl_01.W;;
                        bypassFlow = Fl_02.W;
            }

          }
}
```

```
Element Duct Duct1Left {
 //  Fl_I.MN = 0.56;
   dPqPbase = 0.007;
}

Element VolumeDynamics VolumeLeft3{
ref1 = "Duct1Left.Fl_I";
ref2 = "Fl_O";
LastVolume = "TRUE";
length = 10;
volume = 10*length;

void calculate () {
    //flowSave = VolumeRight45.w1-BurnerRight.Wfuel;
          //pwr = -HPCRight.pwr/C_BTU_PER_SECtoHP;
          dp = Duct1Left.Fl_O.Pt - Duct1Left.Fl_I.Pt;
          VolumeDynamics::calculate();

          }

}

Element Compressor HPCLeft {
   #include "src\D60hpc.ncp";
   real stallSignal = 0;
   S_map.effDes = 0.8223;
   S_map.NcDes = 100.0;
   S_map.PRdes = 6.3665;
   S_map.RlineMap = 2.00;


   InterStageBleedOutPort  Cool1 {
     fracBldWork = 0.66;
     fracBldW = 0.00000;
   }

   InterStageBleedOutPort  Cool2 {
     fracBldWork = 1.0;
     fracBldW = 0.00000;
   }

}

Element Duct Duct2Left {
```

```
 //  Fl_I.MN = 0.40;
     dPqPbase = 0.001;

}


Element Bleed Bld3Left {
   BleedOutPort Cool1;
   BleedOutPort Cool2;
   BleedOutPort CustBld;
   Cool1.fracW = 0.00000;
   Cool2.fracW = 0.00000;
   //Fl_O.Aphy = 23.6;
   void preexecute() {
     CustBld.fracW = 0.0/Fl_I.W;
    }
}


Element FuelStart FuelLeft {
   LHV = 18530;
}


Element Burner BurnerLeft {
   TtCombOut = 2800;
   FAR = 0.02942;
   real wfuelHour = 3.07439*3600;
   wfuelHour = 3.07439*3600/5.0;
   dPqPBase = 0.04;
   effBase = 0.999;
   switchBurn = "FUEL";

      void preexecute() {
     Wfuel = wfuelHour/3600;
    }


     Subelement ThermalMass S_Qhx  {
                    Wdes = 86.2349;
                    kcDes = 9.012543e-6;
                    muDes = 2.40142e-5;
                    massMat = 245.24*(15130.539/41500)/10;
                    CpMat = 0.106045;
                    Ahx = 15;
                    ChxDes = 0.795;
```

```
                ChxDes = 0.467;
                switchForm = "ADD_SOLVER";
                Tmat = 2400;
                wtdAvg_Fl = 0.5;
  }

}


Element Turbine HPTLeft {
  #include "src\D60hpt.ncp";

  FlowStation FS41;

  S_map.parmMap = 3.0439;
  //S_map.parmMap = 3.0439;
  S_map.effDes = 0.87338;
  S_map.parmNcDes = 100;

  InterStageBleedInPort  Non_ChargeableBld {
   Pfract = 1.0;
  }

  InterStageBleedInPort  ChargeableBld {
   Pfract = 0.0;
  }

  Subelement CoolIt Cool {
   cool1 = "Non_ChargeableBld";
   cool2 = "ChargeableBld";
   desVaneTemp[1]  = 2200;
   desBladeTemp[1] = 2200;
   desVaneTemp[2]  = 2200;
   desBladeTemp[2] = 2200;
   coolTypeVane[1] = 5;
   coolTypeBlade[1] = 5;
   coolTypeVane[2] = 5;
   coolTypeBlade[2] = 5;
   nStages = 2;
   xFactor = 0.6;
   xFactor1 = 0.95;
   safety = 150;
  }

  void postexecute() {
   FS41.copyFlow("Fl_I");
```

```
    FS41.add("Non_ChargeableBld");
    Cool.run();
   }
}


Element Turbine LPTLeft {
  #include "src\D60lpt.ncp";

  FlowStation FS49;

  S_map.parmMap = 1.5998;
  S_map.parmMap = 2.0;
  S_map.effDes = 0.94245;
  S_map.parmNcDes = 100.;

  InterStageBleedInPort  Non_ChargeableBld {
   Pfract = 1.0;
  }

  InterStageBleedInPort  ChargeableBld {
   Pfract = 0.0;
  }

  Subelement CoolIt Cool {
   cool1 = "Non_ChargeableBld";
   cool2 = "ChargeableBld";
   desVaneTemp[1]  = 2100;
   desBladeTemp[1] = 2100;
   desVaneTemp[2]  = 2100;
   desBladeTemp[2] = 2100;
   coolTypeVane[1] = 4;
   coolTypeBlade[1] = 4;
   coolTypeVane[2] = 4;
   coolTypeBlade[2] = 4;
   nStages = 2;
   xFactor = 0.7;
   xFactor1 = 1.4;
   safety = 150;
  }

  void postexecute() {
   FS49.copyFlow("Fl_I");
   FS49.add("Non_ChargeableBld");
   Cool.run();
  }
```

```
}


Element Duct Duct3Left {
 //  Fl_I.MN = 0.40;
    dPqPbase = 0.020;

}


Element Duct Duct13Left {
 //  Fl_I.MN = 0.30;
    dPqPbase = 0.012;

}

Element VolumeDynamics VolumeLeft13{
ref1 = "Duct13Left.Fl_I";
ref2 = "Fl_O";
//LastVolume = "TRUE";
length = 500;
volume = 100*length;

void calculate () {
   flowSave = VolumeLeft16.Fl_Itemp.W;
          pwr = -0;
          dp = Duct13Left.Fl_O.Pt - Duct13Left.Fl_I.Pt;
          VolumeDynamics::calculate();

          }

}

Element Duct Duct4Left {
 //  Fl_I.MN = 0.40;
}



Element Nozzle NozzleLeft {

  switchCoef="CV"; Cv=0.9524;
  PsExhName="AmbientLeft.Fl_O.Ps";
  real AthColdControl = 178.17*144/6;
  //AthCold = 340;
    void preexecute() {
```

```
        AthCold = AthColdControl/144;
    }

}


Element FlowEnd Nozz_EndLeft {
}


Element FlowEnd OBsink3Left { }


Element Shaft1 HP_SHAFTLeft {
  ShaftInputPort HPC, HPT;

 Nmech=15130.539;
 Nmech=41500;
 inertia = 3.379*(15130.539/41500)**4;
 HPX=200;
}


Element Shaft1 LP_SHAFTLeft {
  ShaftInputPort FAN, LPT;

 Nmech = 11433.539;
 Nmech = 18000;
 inertia = 4.2499*(11433.539/18000)**4;
}

//-------------------------------------------------------------------------
//                        Right Engine
//-------------------------------------------------------------------------

Element FlightConditions AmbientRight {
  alt = 0;
  MN = 0.00;
  W = 209.0;
  W = 80.0;
  void preexecute() {

                if(VolumeRight13.switchDyn == "ON" && switchDes == OFFDESIGN)
{
                        W = VolumeRight1.Fl_Itemp.W;
            }
```

```
        }
}


Element Inlet InletRight {

  real zepr;
  real WC_Schedule, WC_DESIGN;
  WC_DESIGN = 400.00;
  WC_DESIGN = 80.00;
//   W = 380.00;

}

Element VolumeDynamics VolumeRight1{
ref1 = "InletRight.Fl_I";
ref2 = "Fl_O";
length = 10;
volume = 100*length;
//LastVolume = "TRUE";

void calculate () {
   flowSave = VolumeRight2.Fl_Itemp.W;
  // flowSave = VolumeRight13.Fl_Itemp.W + SplitterRight.coreFlow ;
   dp = InletRight.Fl_O.Pt - InletRight.Fl_I.Pt;
        VolumeDynamics::calculate();

        }

}


Element Compressor FanRight {
  #include "src\D60fan.ncp";
  real stallSignal = 0;
  S_map.effDes = 0.8445;
  S_map.PRdes = 2.303;
  S_map.RlineMap = 2.10;
  S_map.NcDes = 1.0;    // set to 1.0 in ncp file R. Denney 6-21-05

}

Element VolumeDynamics VolumeRight2{
ref1 = "FanRight.Fl_I";
ref2 = "Fl_O";
//LastVolume = "TRUE";
```

```
length = 10;
volume = 100*length;

void calculate () {
   flowSave = VolumeRight3.Fl_Itemp.W + VolumeRight13.Fl_Itemp.W;
        pwr =
(FanRight.WcCalc/FanRight.Fl_I.Wc)*FanRight.pwr/C_BTU_PER_SECtoHP;
        //pwr = FanRight.pwr/C_BTU_PER_SECtoHP;
        dp = FanRight.Fl_O.Pt - FanRight.Fl_I.Pt;
        VolumeDynamics::calculate();
        }

}

Element Splitter SplitterRight {
  BPR = 1.0;
  BPR = 4.0;
  real coreFlow, bypassFlow;
  void preexecute() {

                if(VolumeRight13.switchDyn == "ON" && switchDes == OFFDESIGN)
{
//                      BPR = VolumeRight13.Fl_Itemp.W/coreFlow;
                        BPR = VolumeRight13.Fl_Itemp.W/VolumeRight3.Fl_Itemp.W;

            }

   }


   void postexecute() {
        //cout<<"SplitterRight"<<endl;

                if(VolumeRight13.switchDyn == "OFF" ) {
                        coreFlow = Fl_01.W;
                        bypassFlow = Fl_02.W;
            }

          }

}


Element Duct Duct1Right {
 //  Fl_I.MN = 0.56;
  dPqPbase = 0.007;
```

```
}

Element VolumeDynamics VolumeRight3{
ref1 = "Duct1Right.Fl_I";
ref2 = "Fl_O";
LastVolume = "TRUE";
length = 10;
volume = 10*length;

void calculate () {
   //flowSave = VolumeRight45.w1-BurnerRight.Wfuel;
         //pwr = -HPCRight.pwr/C_BTU_PER_SECtoHP;
         dp = Duct1Right.Fl_O.Pt - Duct1Right.Fl_I.Pt;
         VolumeDynamics::calculate();

         }

}

Element Compressor HPCRight {
  #include "src\D60hpc.ncp";
  real stallSignal = 0;
  S_map.effDes = 0.8223;
  S_map.NcDes = 100.0;
  S_map.PRdes = 6.3665;
  S_map.RlineMap = 2.00;


  InterStageBleedOutPort  Cool1 {
    fracBldWork = 0.66;
    fracBldW = 0.00000;
  }

  InterStageBleedOutPort  Cool2 {
    fracBldWork = 1.0;
    fracBldW = 0.00000;
  }

}


Element Duct Duct2Right {
//   Fl_I.MN = 0.40;
     dPqPbase = 0.001;

}
```

```
Element Bleed Bld3Right {
  BleedOutPort Cool1;
  BleedOutPort Cool2;
  BleedOutPort CustBld;
  Cool1.fracW = 0.00000;
  Cool2.fracW = 0.00000;
  Fl_O.Aphy = 23.6;
  void preexecute() {
   CustBld.fracW = 0.0/Fl_I.W;
  }
}


Element FuelStart FuelRight {
  LHV = 18530;
}


Element Burner BurnerRight {
  TtCombOut = 2800;
  FAR = 0.02942;
  real wfuelHour = 3.07439*3600/5.0;
  dPqPBase = 0.04;
  effBase = 0.999;
  switchBurn = "FUEL";

    void preexecute() {
   Wfuel = wfuelHour/3600;
  }


   Subelement ThermalMass S_Qhx  {
                Wdes = 86.2349;
                kcDes = 9.012543e-6;
                muDes = 2.40142e-5;
                massMat = 245.24*(15130.539/41500)/10;
                CpMat = 0.106045;
                Ahx = 15;
                ChxDes = 0.795;
                ChxDes = 0.467;
                switchForm = "ADD_SOLVER";
                Tmat = 2400;
                wtdAvg_Fl = 0.5;
 }
```

```
}


Element Turbine HPTRight {
  #include "src\D60hpt.ncp";

  FlowStation FS41;

  S_map.parmMap = 3.0439;
  S_map.parmMap = 3.0439;
  S_map.effDes = 0.87338;
  S_map.parmNcDes = 100;

  InterStageBleedInPort  Non_ChargeableBld {
   Pfract = 1.0;
  }

  InterStageBleedInPort  ChargeableBld {
   Pfract = 0.0;
  }

  Subelement CoolIt Cool {
   cool1 = "Non_ChargeableBld";
   cool2 = "ChargeableBld";
   desVaneTemp[1]  = 2200;
   desBladeTemp[1] = 2200;
   desVaneTemp[2]  = 2200;
   desBladeTemp[2] = 2200;
   coolTypeVane[1] = 5;
   coolTypeBlade[1] = 5;
   coolTypeVane[2] = 5;
   coolTypeBlade[2] = 5;
   nStages = 2;
   xFactor = 0.6;
   xFactor1 = 0.95;
   safety = 150;
  }

  void postexecute() {
   FS41.copyFlow("Fl_I");
   FS41.add("Non_ChargeableBld");
   Cool.run();
  }
}
```

```
Element Turbine LPTRight {
  #include "src\D60lpt.ncp";


  FlowStation FS49;

  S_map.parmMap = 1.5998;
  S_map.parmMap = 2.0;
  S_map.effDes = 0.94245;
  S_map.parmNcDes = 100.;

  InterStageBleedInPort  Non_ChargeableBld {
   Pfract = 1.0;
  }

  InterStageBleedInPort  ChargeableBld {
   Pfract = 0.0;
  }

  Subelement CoolIt Cool {
   cool1 = "Non_ChargeableBld";
   cool2 = "ChargeableBld";
   desVaneTemp[1]  = 2100;
   desBladeTemp[1] = 2100;
   desVaneTemp[2]  = 2100;
   desBladeTemp[2] = 2100;
   coolTypeVane[1] = 4;
   coolTypeBlade[1] = 4;
   coolTypeVane[2] = 4;
   coolTypeBlade[2] = 4;
   nStages = 2;
   xFactor = 0.7;
   xFactor1 = 1.4;
   safety = 150;
  }

  void postexecute() {
   FS49.copyFlow("Fl_I");
   FS49.add("Non_ChargeableBld");
   Cool.run();
  }
}


Element Duct Duct3Right {
```

```
//   Fl_I.MN = 0.40;
     dPqPbase = 0.020;

//   }
}


Element Duct Duct13Right {
//   Fl_I.MN = 0.30;
    dPqPbase = 0.012;
}

Element VolumeDynamics VolumeRight13{
ref1 = "Duct13Right.Fl_I";
ref2 = "Fl_O";
//LastVolume = "TRUE";
length = 500;
volume = 100*length;

void calculate () {
//cout<<"Right13"<<endl;
    flowSave = VolumeRight16.Fl_Itemp.W;
          pwr = -0;
          dp = Duct13Right.Fl_O.Pt - Duct13Right.Fl_I.Pt;
          if(time> 0.01) {
                    //cout<<"Fl_Itemp.W = "<<Fl_Itemp.W<<endl;
                     }
          VolumeDynamics::calculate();
                    if(time> 0.01) {
          //cout<<"Fl_Itemp.W = "<<Fl_Itemp.W<<endl;
          //cout<<"dWqdt = "<<dWqdt<<endl;
          }
          }

}


Element Duct Duct4Right {
 //  Fl_I.MN = 0.40;
}

Element Nozzle NozzleRight {
//   Fl_I.MN = 0.40;
 //  switchType = "CON_DIV";
 //  switchCoef = "CV";
 //  Cv = 0.975;
```

```
  switchCoef="CV"; Cv=0.9524;  //ICLS prediction pg 513
  PsExhName="AmbientRight.Fl_O.Ps";
  real AthColdControl = 178.17*144/6;
  //AthCold = 340;
     void preexecute() {
        AthCold = AthColdControl/144;
     }
}


Element FlowEnd Nozz_EndRight {
}
Element FlowEnd OBsink3Right { }


Element Shaft1 HP_SHAFTRight {
   ShaftInputPort HPC, HPT;

  Nmech=15130.539;
  Nmech=41500;
  inertia = 3.379*(15130.539/41500)**4;
  HPX=200;
}


Element Shaft1 LP_SHAFTRight {
   ShaftInputPort FAN, LPT;

  Nmech = 11433.539;
  Nmech = 18000;
  inertia = 4.2499*(11433.539/18000)**4;
}

//-------------------------------------------------------------------------
//                        Tip-Jet Reaction Drive
//-------------------------------------------------------------------------
Element Duct Duct16Left {
   Fl_I.MN = 0.30;
//     dPqPbase = 0.012;
   Subelement dPqPMach S_dP {
     dPqPMNdes = 0.012;
     expMN = 1.75;
   }
}

Element VolumeDynamics VolumeLeft16{
```

```
ref1 = "Duct16Left.Fl_I";
ref2 = "Fl_O";
LastVolume = "TRUE";
length = 500;
volume = 100*length;

void calculate () {
//cout<<"Left16"<<endl;
   //flowSave = SplitterLeft.Fl_I.W;
          pwr = -0;
          dp = Duct16Left.Fl_O.Pt - Duct16Left.Fl_I.Pt;
          VolumeDynamics::calculate();

          }

}

Element Duct Duct16Right {
  Fl_I.MN = 0.30;
//    dPqPbase = 0.012;
  Subelement dPqPMach S_dP {
    dPqPMNdes = 0.012;
    expMN = 1.75;
  }
}

Element VolumeDynamics VolumeRight16{
ref1 = "Duct16Right.Fl_I";
ref2 = "Fl_O";
LastVolume = "TRUE";
length = 500;
volume = 100*length;

void calculate () {
//cout<<"Right16"<<endl;
   //flowSave = SplitterRight.Fl_I.W;
          pwr = -0;
          dp = Duct16Right.Fl_O.Pt - Duct16Right.Fl_I.Pt;
          VolumeDynamics::calculate();

          }

}

Element Mixer Hub {      // change from Mixer2 R. Denney 6-13-05
  real Area;
```

```
  real Aout = 640.76 ;
  real AreaPriDes, AreaSecDes;
  Fl_I1.MN = 0.30;
//   Cmixer = 0.95;

  int veryFirstPass = TRUE;
  void preexecute() {
    if (veryFirstPass) {
//      PtOut = Fl_I1.Pt;
      veryFirstPass = FALSE;
    }
  }

  void postexecute() {
    Area = Fl_I1.Aphy + Fl_I2.Aphy;
    if (switchDes==DESIGN) {
      AreaPriDes = Fl_I1.Aphy;
      AreaSecDes = Fl_I2.Aphy;
                  }
  }
}

Element Duct Duct91 {
  Fl_I.MN = 0.30;
//    dPqPbase = 0.012;
  Subelement dPqPMach S_dP {
    dPqPMNdes = 0.012;
    expMN = 1.75;
  }
}

Element Duct Duct93 {
  Fl_I.MN = 0.30;
//    dPqPbase = 0.012;
  Subelement dPqPMach S_dP {
    dPqPMNdes = 0.012;
    expMN = 1.75;
  }
}

Element Duct Duct95 {
  Fl_I.MN = 0.30;
//    dPqPbase = 0.012;
  Subelement dPqPMach S_dP {
    dPqPMNdes = 0.012;
    expMN = 1.75;
```

```
  }
}

Element FuelStart TipFuel {
  LHV = 18530;
}


Element Burner TipBurner {

  TtCombOut = 2800;
  FAR = 0.02942;
  real wfuelHour = 7.15684*3600*1.4;
  dPqPBase = 0.04;
  effBase = 0.999;
  switchBurn = "FUEL";

    void preexecute() {
   Wfuel = wfuelHour/3600;
  }


   Subelement ThermalMass S_Qhx  {
                Wdes = 86.2349;
                kcDes = 9.012543e-6;
                muDes = 2.40142e-5;
                massMat = 245.24/100;
                CpMat = 0.106045;
                Ahx = 15;
                ChxDes = 0.795;
                ChxDes = 0.467;
                switchForm = "ADD_SOLVER";
                Tmat = 2400;
                wtdAvg_Fl = 0.5;
 }


}

Element Nozzle TipNozzle {
//   Fl_I.MN = 0.40;
 //  switchType = "CON_DIV";
 //  switchCoef = "CV";
 //  Cv = 0.975;
  real torque = 2000;
  real power = 1000;
```

```
  real V_tip;
  real rotorRadius = 1;
  ShaftOutputPort Sh_O {
    description = "Mechanical connection to the shaft";
  }

  switchCoef="CV"; Cv=0.9524;  //ICLS prediction pg 513
  PsExhName="AmbientRight.Fl_O.Ps";
  real AthColdControl = 668.13*144/4;
  //real AthColdControlInput = 340;
  //AthCold = 340;
     void preexecute() {
        AthCold = AthColdControl/(144/4);
     }


     void postexecute() {
           //cout<<"TipNozzle"<<endl;

             V_tip = 2*3.12415*Sh_O.Nmech*rotorRadius/60;
             torque = Fl_O.W*(Fl_O.V-V_tip)/32.2;
             power = torque*Sh_O.Nmech/C_HP_PER_RPMtoFT_LBF;
             Sh_O.trq = torque;

           }
}

Element FlowEnd Bypass_End {
}

Element Element RotorTorqueDemand {

          Table FlowOut( real time ) {
                    time = {   0.0,   0.5,  0.51,    50.0  }
                    FlowOut = { 1, 1.01, 1.05 , 1.05  }
          }
  real torque =  -5000;
  real torqueBase = torque;

  ShaftOutputPort Sh_O {
    description = "Mechanical connection to the shaft";
  }

  void calculate() {
       //torque = torqueBase*FlowOut(time);
             Sh_O.trq = torque;
```

```
  }
}


Element Shaft1 TipJetRotor {
   ShaftInputPort Demand, TipJet;

  Nmech = 11433.539;
  inertia = 4.2499/5;
  inertia = 4.2499*2;
}
//-------------------------------------------------------------------------
//                        System Performance
//-------------------------------------------------------------------------



Element EngPerf Eng {

   real  A[][][] ;

   real FnFullPower, ThrustTarget, CEPRRight, CEPRLeft, ETR, EPRRight, EPRLeft,
EPRTip, PC, OPR_Jon, PLA;
   real FnRight, FnLeft, TSFCRight, TSFCLeft;
   real EPRtarget, SpeedTarget, FA, FA_speed, stallSignal, ETRdemand,
EPRRightDemand, EPRLeftDemand, EPRTipDemand, CEPRRightDemand,
CEPRLeftDemand;
   real Fnet_local, FgInstalled, FnetInstalled, TSFCinstalled, Q, Fdrag = 0.0;
   real beta, A9QA10, L = 100.0, r10 = 30.12, A10 = PI * r10**2;
   real TipJetThrust, RotorThrustDemand;

         Table TB_EPR( real NcorrMap) {

                        NcorrMap = {20,       27.5 ,   35,       42.5 ,  50 }
                        ghllMap = {  1.17042, 1.43701, 1.7022, 1.96001, 2.21497}
     NcorrMap.interp = "linear" ;
     NcorrMap.extrap = "none" ;
         }

         Table TB_ETR( real NcorrMap) {

                        NcorrMap = {20,       27.5 ,   35,       42.5 ,  50 }
                        ghllMap = {  3.36892, 3.664193, 3.959465, 4.254738, 4.55001}
     NcorrMap.interp = "linear" ;
     NcorrMap.extrap = "none" ;
         }
```

```
  #include <src\nozzle.cdmap>;

  void postexecute() {
 //cout<<"ENG = "<<LPTRight.S_map.parmMap<<endl;
  //cout<<"ENG = "<<LPTLeft.S_map.parmMap<<endl;
  //CEPRRightDemand = 15391.4*FlowOut(time);
        //CEPRLeftDemand = 15391.4*FlowOut(time);
  EPRRight = Duct4Right.Fl_O.Pt / FanRight.Fl_I.Pt ;
  EPRLeft = Duct4Left.Fl_O.Pt / FanLeft.Fl_I.Pt ;
  EPRTip = Hub.Fl_O.Pt / FanLeft.Fl_I.Pt ;
  CEPRRight = Duct4Right.Fl_O.Pt/ HPCRight.Fl_I.Pt;
        CEPRLeft = Duct4Left.Fl_O.Pt/ HPCLeft.Fl_I.Pt;
  FnRight = NozzleRight.Fg - InletRight.Fram;   //EngPerf doesn't recognize
InstalledInlet
        FnLeft = NozzleLeft.Fg - InletLeft.Fram;   //EngPerf doesn't recognize
InstalledInlet
  TSFCRight = BurnerRight.Wfuel*3600 / NozzleRight.Fg ;
        TSFCLeft = BurnerLeft.Wfuel*3600 / NozzleLeft.Fg ;
        //cout<<"Duct4Right.Fl_O.Pt = "<<Duct4Right.Fl_O.Pt<<endl;
        //cout<<"Duct4Left.Fl_O.Pt = "<<Duct4Left.Fl_O.Pt<<endl;
        //cout<<"CEPRRight = "<<CEPRRight<<endl;

  }
}

#include <src\linearModels.inp>;


//---------------------------------------------------------------------------
//      linkPorts - Left Engine
//---------------------------------------------------------------------------

linkPorts( "AmbientLeft.Fl_O"      , "InletLeft.Fl_I"         , "FS0Left"   );
linkPorts( "InletLeft.Fl_O"        , "VolumeLeft1.Fl_I"       , "VS2Left"   );
linkPorts( "VolumeLeft1.Fl_O"      , "FanLeft.Fl_I"           , "FS2Left"   );
linkPorts( "FanLeft.Fl_O"          , "VolumeLeft2.Fl_I"       , "VS21Left"   );
linkPorts( "VolumeLeft2.Fl_O"      , "SplitterLeft.Fl_I"      , "FS21Left"  );
linkPorts( "SplitterLeft.Fl_01"    , "Duct1Left.Fl_I"         , "FS23Left"  );
linkPorts( "Duct1Left.Fl_O"        , "VolumeLeft3.Fl_I"       , "VS25Left"   );
linkPorts( "VolumeLeft3.Fl_O"    , "HPCLeft.Fl_I"           , "FS25Left"  );

linkPorts( "HPCLeft.Fl_O"          , "Duct2Left.Fl_I"         , "FS3Left"   );
linkPorts( "Duct2Left.Fl_O"        , "Bld3Left.Fl_I"          , "FS31Left"  );
linkPorts( "Bld3Left.Fl_O"         , "BurnerLeft.Fl_I"        , "FS36Left"  );
linkPorts( "FuelLeft.Fu_O"         , "BurnerLeft.Fu_I"        , "FuelInLeft");
linkPorts( "BurnerLeft.Fl_O"       , "HPTLeft.Fl_I"           , "FS4Left"   );
```

```
linkPorts( "HPTLeft.Fl_O"          , "LPTLeft.Fl_I"           , "FS48Left"  );
linkPorts( "LPTLeft.Fl_O"          , "Duct3Left.Fl_I"         , "FS5Left"   );


// BYPASS linkPorts
linkPorts( "SplitterLeft.Fl_02"    , "Duct13Left.Fl_I"        , "FS13Left"  );

// Mixed stream
linkPorts( "Duct3Left.Fl_O"   , "Duct4Left.Fl_I"          , "FS6Left"   );

linkPorts( "Duct4Left.Fl_O"       , "NozzleLeft.Fl_I"        , "FS7Left"   );
linkPorts( "NozzleLeft.Fl_O"       , "Nozz_EndLeft.Fl_I"      , "FS9Left"   );


// BLEED linkPorts
linkPorts("HPCLeft.Cool1"          , "LPTLeft.Non_ChargeableBld", "C_LPTinltLeft" );
linkPorts("HPCLeft.Cool2"          , "LPTLeft.ChargeableBld"    , "C_LPTexitLeft" );
linkPorts("Bld3Left.Cool1"         , "HPTLeft.Non_ChargeableBld", "C_HPTinltLeft" );
linkPorts("Bld3Left.Cool2"         , "HPTLeft.ChargeableBld"    , "C_HPTexitLeft" );
linkPorts("Bld3Left.CustBld"       , "OBsink3Left.Fl_I"         , "FS_CBLeft"    );

// SHAFT linkPorts
linkPorts( "FanLeft.Sh_O"          , "LP_SHAFTLeft.FAN"       , "FANworkLeft" );
linkPorts( "LPTLeft.Sh_O"          , "LP_SHAFTLeft.LPT"       , "LPTworkLeft" );
linkPorts( "HPTLeft.Sh_O"          , "HP_SHAFTLeft.HPT"       , "HPTworkLeft" );
linkPorts( "HPCLeft.Sh_O"          , "HP_SHAFTLeft.HPC"       , "HPCworkvLeft" );

//--------------------------------------------------------------------------
//       linkPorts - Right Engine
//--------------------------------------------------------------------------

linkPorts( "AmbientRight.Fl_O"       , "InletRight.Fl_I"         , "FS0Right"   );
linkPorts( "InletRight.Fl_O"         , "VolumeRight1.Fl_I"       , "VS2Right"   );
linkPorts( "VolumeRight1.Fl_O"       , "FanRight.Fl_I"           , "FS2Right"   );
linkPorts( "FanRight.Fl_O"           , "VolumeRight2.Fl_I"       , "VS21Right"   );
linkPorts( "VolumeRight2.Fl_O"       , "SplitterRight.Fl_I"      , "FS21Right"  );

linkPorts( "SplitterRight.Fl_01"     , "Duct1Right.Fl_I"         , "FS23Right"  );
linkPorts( "Duct1Right.Fl_O"         , "VolumeRight3.Fl_I"       , "VS25Right"   );
linkPorts( "VolumeRight3.Fl_O"     , "HPCRight.Fl_I"           , "FS25Right"  );

linkPorts( "HPCRight.Fl_O"           , "Duct2Right.Fl_I"         , "FS3Right"   );
linkPorts( "Duct2Right.Fl_O"         , "Bld3Right.Fl_I"          , "FS31Right"  );
linkPorts( "Bld3Right.Fl_O"          , "BurnerRight.Fl_I"        , "FS36Right"  );
linkPorts( "FuelRight.Fu_O"          , "BurnerRight.Fu_I"        , "FuelInRight");
linkPorts( "BurnerRight.Fl_O"        , "HPTRight.Fl_I"           , "FS4Right"   );
```

```
linkPorts( "HPTRight.Fl_O"          , "LPTRight.Fl_I"           , "FS48Right"  );
linkPorts( "LPTRight.Fl_O"          , "Duct3Right.Fl_I"         , "FS5Right"   );

// BYPASS linkPorts
linkPorts( "SplitterRight.Fl_02"    , "Duct13Right.Fl_I"        , "FS13Right"  );

// Mixed stream
linkPorts( "Duct3Right.Fl_O"    , "Duct4Right.Fl_I"         , "FS6Right"   );

linkPorts( "Duct4Right.Fl_O"        , "NozzleRight.Fl_I"        , "FS7Right"   );
linkPorts( "NozzleRight.Fl_O"       , "Nozz_EndRight.Fl_I"      , "FS9Right"   );


// BLEED linkPorts
linkPorts("HPCRight.Cool1"          , "LPTRight.Non_ChargeableBld", "C_LPTinltRight"
);
linkPorts("HPCRight.Cool2"          , "LPTRight.ChargeableBld"    , "C_LPTexitRight" );
linkPorts("Bld3Right.Cool1"         , "HPTRight.Non_ChargeableBld", "C_HPTinltRight"
);
linkPorts("Bld3Right.Cool2"         , "HPTRight.ChargeableBld"    , "C_HPTexitRight" );
linkPorts("Bld3Right.CustBld"       , "OBsink3Right.Fl_I"         , "FS_CBRight"    );

// SHAFT linkPorts
linkPorts( "FanRight.Sh_O"          , "LP_SHAFTRight.FAN"       , "FANworkRight" );
linkPorts( "LPTRight.Sh_O"          , "LP_SHAFTRight.LPT"       , "LPTworkRight" );
linkPorts( "HPTRight.Sh_O"          , "HP_SHAFTRight.HPT"       , "HPTworkRight" );
linkPorts( "HPCRight.Sh_O"          , "HP_SHAFTRight.HPC"       , "HPCworkvRight"
);

//-------------------------------------------------------------------------
//      linkPorts - TipJet Reaction Drive
//-------------------------------------------------------------------------
//linkPorts( "SplitterBypassLeft.Fl_01"
linkPorts( "Duct13Left.Fl_O"       , "VolumeLeft13.Fl_I"       , "VS13Left"   );
linkPorts( "VolumeLeft13.Fl_O"     , "Duct16Left.Fl_I"         , "FS16Left"  );
linkPorts( "Duct16Left.Fl_O"       , "VolumeLeft16.Fl_I"       , "VS16Left"   );
linkPorts( "VolumeLeft16.Fl_O"     , "Hub.Fl_I2"               , "FS19Left"  );

//linkPorts( "SplitterBypassRight.Fl_01"
linkPorts( "Duct13Right.Fl_O"        , "VolumeRight13.Fl_I"       , "VS13Right"   );
linkPorts( "VolumeRight13.Fl_O"      , "Duct16Right.Fl_I"         , "FS16Right"  );
linkPorts( "Duct16Right.Fl_O"        , "VolumeRight16.Fl_I"       , "VS16Right"   );
linkPorts( "VolumeRight16.Fl_O"      , "Hub.Fl_I1"         , "FS19Right"  );

linkPorts( "Hub.Fl_O"                  ,  "Duct91.Fl_I"        , "FS91"  );
linkPorts( "Duct91.Fl_O"               ,  "Duct93.Fl_I"        , "FS93"  );
```

```
linkPorts( "Duct93.Fl_O"                ,  "Duct95.Fl_I"         , "FS95"  );
linkPorts( "Duct95.Fl_O"                , "TipBurner.Fl_I"         , "FS97"  );
linkPorts( "TipFuel.Fu_O"               , "TipBurner.Fu_I"         , "TipFuelIn");
linkPorts( "TipBurner.Fl_O"             , "TipNozzle.Fl_I"         , "FS98"  );
linkPorts( "TipNozzle.Fl_O"             , "Bypass_End.Fl_I"         , "FS99"  );

linkPorts( "TipNozzle.Sh_O"           , "TipJetRotor.TipJet"        , "TipJetTorque" );
linkPorts( "RotorTorqueDemand.Sh_O"           , "TipJetRotor.Demand"        ,
"DemandTorque" );
```

APPENDIX B

NON-MODEL BASED RELATIVE GAIN ARRAY

Table 37: Relative Gain Array of Tip-Jet Reaction Drive System at 0.1 rad/s

Relative Gain Array	Right Fuel Flow	Right Exhaust Area	Left Fuel Flow	Left Exhaust Area	Tip-Jet Fuel Flow	Tip-Jet Exhaust Area
LP Shaft Right	**0.356**	**0.204**	0.000	0.000	0.011	0.070
HP Shaft Right	**0.292**	0.027	0.000	0.000	0.006	0.019
EPR Right	**0.250**	**0.261**	0.000	0.000	-0.017	0.062
CEPR Right	0.040	**0.480**	0.000	0.000	-0.034	0.081
LP Shaft Left	0.000	0.000	**0.355**	**0.205**	0.011	0.070
HP Shaft Left	0.000	0.000	**0.292**	0.027	0.006	0.019
EPR Left	0.000	0.000	**0.251**	**0.260**	-0.017	0.062
CEPR Left	0.000	0.000	0.040	**0.480**	-0.034	0.081
Tip Jet Shaft	0.001	-0.001	0.001	-0.002	**1.374**	**-0.372**
EPR Tip	0.016	0.007	0.015	0.007	**-0.106**	**0.321**
Duct Pressure Right	0.044	0.025	0.002	-0.001	**-0.100**	**0.294**
Duct Pressure Left	0.002	-0.001	0.043	0.025	**-0.100**	**0.294**

Table 38: Relative Gain Array of Tip-Jet Reaction Drive System at 1 rad/s

Relative Gain Array	Right Fuel Flow	Right Exhaust Area	Left Fuel Flow	Left Exhaust Area	Tip-Jet Fuel Flow	Tip-Jet Exhaust Area
LP Shaft Right	**0.359**	**0.202**	0.000	0.000	0.010	0.061
HP Shaft Right	**0.306**	0.027	0.000	0.000	0.005	0.016
EPR Right	**0.242**	**0.267**	0.000	0.000	-0.019	0.067
CEPR Right	0.041	**0.478**	0.000	-0.001	-0.033	0.082
LP Shaft Left	0.000	0.000	**0.358**	**0.202**	0.010	0.061
HP Shaft Left	0.000	0.000	**0.307**	0.027	0.005	0.016
EPR Left	0.000	0.000	**0.242**	**0.266**	-0.019	0.067
CEPR Left	0.000	-0.001	0.041	**0.478**	-0.033	0.082
Tip Jet Shaft	0.000	0.000	0.000	-0.001	**1.392**	**-0.391**
EPR Tip	0.013	0.006	0.013	0.006	**-0.110**	**0.332**
Duct Pressure Right	0.040	0.023	0.000	-0.002	**-0.104**	**0.304**
Duct Pressure Left	0.000	-0.002	0.040	0.024	**-0.104**	**0.304**

Table 39: Relative Gain Array of Tip-Jet Reaction Drive System at 10 rad/s

Relative Gain Array	Right Fuel Flow	Right Exhaust Area	Left Fuel Flow	Left Exhaust Area	Tip-Jet Fuel Flow	Tip-Jet Exhaust Area
LP Shaft Right	**0.349**	**0.169**	0.000	0.000	-0.002	0.006
HP Shaft Right	**0.429**	0.034	0.000	0.000	0.001	-0.002
EPR Right	**0.175**	**0.325**	-0.004	-0.002	-0.001	0.070
CEPR Right	0.045	**0.470**	-0.002	-0.001	-0.003	0.055
LP Shaft Left	0.000	0.000	**0.347**	**0.169**	-0.001	0.006
HP Shaft Left	0.000	0.000	**0.432**	0.034	0.001	-0.002
EPR Left	-0.003	-0.002	**0.175**	**0.324**	-0.001	0.070
CEPR Left	-0.003	-0.001	0.045	**0.471**	-0.002	0.055
Tip Jet Shaft	0.000	0.000	0.000	0.000	**1.541**	**-0.542**
EPR Tip	-0.002	-0.001	-0.002	-0.001	**-0.193**	**0.458**
Duct Pressure Right	0.021	0.012	-0.011	-0.006	**-0.170**	**0.413**
Duct Pressure Left	-0.012	-0.006	0.021	0.012	**-0.171**	**0.414**

APPENDIX C

MODEL BASED RELATIVE GAIN ARRAY

Table 40: Relative Gain Array of Tip-Jet Reaction Drive System Model Based Control at 0.1 rad/s

Relative Gain Array	Right Fuel Flow	Right Exhaust Area	Left Fuel Flow	Left Exhaust Area	Tip-Jet Fuel Flow	Exhaust Area
Thrust Right	**0.5494**	0.0252	0.0123	0.0124	-0.0486	**0.0681**
LP Shaft Right	**0.0538**	0.0481	-0.004	-0.0041	0.0176	-0.0218
HP Shaft Right	**0.0515**	0.0042	-0.0014	-0.0014	0.006	-0.0074
Fan Stall Right	**0.206**	**0.1775**	**0.0596**	**0.0606**	**-0.0814**	**0.5374**
HPC Stall Right	0.0411	**0.0508**	0.0025	0.0025	-0.0076	0.0116
CEPR Right	-0.0033	**0.5912**	0.0107	0.0109	-0.0464	**0.0587**
Thrust Right	0.0123	0.0125	**0.5496**	0.0252	-0.0486	**0.0681**
LP Shaft Left	-0.004	-0.0041	**0.0538**	0.0481	0.0176	-0.0218
HP Shaft Left	-0.0014	-0.0014	**0.0516**	0.0042	0.006	-0.0074
Fan Stall Left	**0.0597**	**0.0606**	**0.2061**	**0.1774**	**-0.0813**	**0.5373**
HPC Stall Left	0.0025	0.0025	0.0408	**0.0509**	-0.0078	0.012
CEPR Left	0.0107	0.0109	-0.0033	**0.5912**	-0.0465	**0.0587**
TipJetRotor Shaft	-0.0004	-0.0006	-0.0004	-0.0007	**1.3869**	**-0.3849**
Hub Pressure	0.0221	0.0226	0.0221	0.0226	**-0.0658**	**0.0913**

Table 41: Relative Gain Array of Tip-Jet Reaction Drive System Model Based Control at 1 rad/s

Relative Gain Array	Right Fuel Flow	Right Exhaust Area	Left Fuel Flow	Left Exhaust Area	Tip-Jet Fuel Flow	Exhaust Area
Thrust Right	**0.4573**	**0.0592**	0.0094	0.0101	-0.0352	**0.0525**
LP Shaft Right	0.0414	0.041	-0.0023	-0.0028	0.0132	-0.0161
HP Shaft Right	0.0414	0.0034	-0.0008	-0.001	0.0045	-0.0055
Fan Stall Right	**0.1843**	**0.1711**	**0.0581**	**0.0668**	**-0.1287**	**0.5907**
HPC Stall Right	**0.1816**	0.0439	0.0018	0.0006	0.0109	-0.0093
CEPR Right	0.0035	**0.5797**	0.0078	0.0093	-0.0407	**0.0518**
Thrust Right	0.0093	0.0101	**0.4575**	**0.0592**	-0.0353	**0.0527**
LP Shaft Left	-0.0023	-0.0028	0.0414	0.041	0.0132	-0.0161
HP Shaft Left	-0.0008	-0.001	0.0414	0.0034	0.0045	-0.0055
Fan Stall Left	**0.0582**	**0.0668**	**0.1843**	**0.171**	**-0.1293**	**0.5913**
HPC Stall Left	0.0018	0.0006	**0.1814**	0.044	0.0102	-0.0086
CEPR Left	0.0078	0.0093	0.0034	**0.5797**	-0.0407	**0.0517**
TipJetRotor Shaft	-0.0003	-0.0004	-0.0003	-0.0004	**1.4073**	**-0.4066**
Hub Pressure	0.0169	0.0191	0.0168	0.0191	**-0.0539**	**0.0771**

Table 42: Relative Gain Array of Tip-Jet Reaction Drive System Model Based Control at 10 rad/s

Thrust Right	**0.0849**	**0.256**	0.0003	0.0008	**1.0604**	**-1.0277**
LP Shaft Right	0.0021	0.0044	0	0	-0.0058	0.0058
HP Shaft Right	0.0025	0.0004	0	0	-0.0013	0.0012
Fan Stall Right	0.0304	0.0456	0.0086	0.0247	**-2.1145**	**2.5885**
HPC Stall Right	**0.8863**	0.0053	0.0003	0.0007	**0.2209**	**-0.2094**
CEPR Right	-0.0156	**0.6606**	0	0.0002	**0.3373**	**-0.3325**
Thrust Right	0.0003	0.0008	**0.0847**	**0.2563**	**1.059**	**-1.0266**
LP Shaft Left	0	0	0.0021	0.0044	-0.0059	0.0058
HP Shaft Left	0	0	0.0025	0.0004	-0.0011	0.0011
Fan Stall Left	0.0086	0.0247	0.0304	0.0456	**-2.189**	**2.6631**
HPC Stall Left	0.0003	0.0007	**0.8864**	0.0053	**0.2195**	**-0.2081**
CEPR Left	0	0.0002	-0.0156	**0.6603**	**0.3358**	**-0.3308**
TipJetRotor Shaft	-0.0001	0	-0.0001	0	**1.4673**	**-0.5326**
Hub Pressure	0.0004	0.0014	0.0004	0.0014	**0.6175**	**-0.5978**

APPENDIX D

EXTENDED KALMAN FILTER S-FUNCTION SOURCE CODE

```
function KFfcnFinal(block)

setup(block);

function setup(block)

block.NumDialogPrms = 0;

block.NumInputPorts = 4;
block.NumOutputPorts = 2;


  %% Register dialog parameter: LMS step size
  block.NumDialogPrms = 7;
%   block.DialogPrmsTunable = {'Tunable'};


block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;
block.InputPort(1).Dimensions = [17, 1]; %x
block.InputPort(2).Dimensions = [7, 1]; %u
block.InputPort(3).Dimensions = [17, 1]; %y
block.InputPort(4).Dimensions = [17, 17]; %p


block.OutputPort(1).Dimensions = [17, 1]; %x
block.OutputPort(2).Dimensions = [17, 17]; %p

%% Set block sample time to inherited
block.SampleTimes = [0.02, 0];

block.RegBlockMethod('SetInputPortSamplingMode',@SetInputPortSamplingMode);
% block.RegBlockMethod('SetInputPortDimensions', @SetInpPortDims);
block.RegBlockMethod('Outputs', @Output);


function SetInputPortSamplingMode(block, idx, fd)
block.InputPort(idx).SamplingMode = fd;
block.InputPort(idx).SamplingMode = fd;

block.OutputPort(1).SamplingMode = fd;
block.OutputPort(2).SamplingMode = fd;
```

```
% function SetInpPortDims(block, idx, di)
% block.InputPort(idx).Dimensions = di;
% di

function Output(block)


x = block.InputPort(1).Data;
uLastMove = block.InputPort(2).Data;
ym = block.InputPort(3).Data;
P = block.InputPort(4).Data;


% MPC object setup
A_discrete = block.DialogPrm(1).Data;
B_discrete = block.DialogPrm(2).Data;
C_discrete = block.DialogPrm(3).Data;
D_discrete =block.DialogPrm(4).Data;
Ts=block.DialogPrm(5).Data;     %Sampling time
Q=block.DialogPrm(6).Data;     %Sampling time
R=block.DialogPrm(7).Data;     %Sampling time
%%

% GC_discrete = ss(A_discrete,B_discrete,C_discrete,D_discrete,Ts);



yb = ones(size(ym));
xb = ones(size(x));
ub = ones(size(uLastMove));


xsave = x;
xd = x-xb;
A_discrete;

ud = uLastMove-ub;
yd = ym-yb;
%%  Prediction for state vector and covariance

xd = A_discrete*xd + B_discrete*ud;
P = A_discrete*P*A_discrete' + Q;
xd;
%
%
```

```
% %%  Compute Kalman Gain Factors
K = P*C_discrete'*inv(C_discrete*P*C_discrete' + R);
cond(K);
%
%
% %%  Correction Based Upon Observations
yhat=C_discrete*xd+D_discrete*ud;

yd-yhat;
xd = xd + K*(yd - yhat);
P = P - K*C_discrete*P;
x = xd+xb;
% x(6:17,1) = xsave(6:17,1);
resid = yd-yhat;

%

block.OutputPort(1).Data = x;
block.OutputPort(2).Data = P;
```

APPENDIX E

DISTRIBUTED MPC S-FUNCTION SOURCE CODE

```
function MPCmodelFcnDistributed(block)

setup(block);

function setup(block)

block.NumInputPorts = 12;
block.NumOutputPorts = 3;

  %% Register dialog parameter: LMS step size
  block.NumDialogPrms = 6;
%   block.DialogPrmsTunable = {'Tunable'};


block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;
% block.InputPort(1).Dimensions = [21, 21];
% block.InputPort(2).Dimensions = [21, 7];
% block.InputPort(3).Dimensions = [19, 21];
% block.InputPort(4).Dimensions = [19, 7];
% block.InputPort(5).Dimensions = [21, 1];
% block.InputPort(6).Dimensions = [1, 19];
% block.InputPort(7).Dimensions = [1, 1];
% block.InputPort(8).Dimensions = [6, 1];
%

% block.OutputPort(2).Dimensions = [6, 60];


%% Set block sample time to inherited
block.SampleTimes = [0.02, 0];

block.RegBlockMethod('SetInputPortSamplingMode',@SetInputPortSamplingMode);
block.RegBlockMethod('SetInputPortDimensions', @SetInpPortDims);
% block.RegBlockMethod('SetOutputPortDimensions', @SetOutPortDims);
block.RegBlockMethod('Outputs', @Output);


function SetInputPortSamplingMode(block, idx, fd)
block.InputPort(idx).SamplingMode = fd;
```

```
block.InputPort(idx).SamplingMode = fd;
%
block.OutputPort(1).SamplingMode = fd;
block.OutputPort(2).SamplingMode = fd;
block.OutputPort(3).SamplingMode = fd;

% function SetOutPortDims(block, idx, di)
% block.OutputPort(idx).Dimensions = di;

function SetInpPortDims(block, idx, di)
block.InputPort(idx).Dimensions = di;
 block.OutputPort(1).Dimensions = [6,1];
 block.OutputPort(2).Dimensions = block.InputPort(7).Dimensions;
 block.OutputPort(3).Dimensions = [19,1];
function Output(block)
%
% A_discrete = block.InputPort(1).Data;
% B_discrete = block.InputPort(2).Data;
% C_discrete = block.InputPort(3).Data;
% D_discrete = block.InputPort(4).Data;
% x = block.InputPort(5).Data;
 y = block.InputPort(1).Data;
 r = block.InputPort(2).Data;
 v = block.InputPort(3).Data;
 uLast = block.InputPort(4).Data;
 x = block.InputPort(5).Data;
 pdm = block.InputPort(6).Data;
 uOptLast = block.InputPort(7).Data;
 kRight =  block.InputPort(8).Data;
 kLeft =  block.InputPort(9).Data;
 kTip =  block.InputPort(10).Data;
  maxIter =  block.InputPort(11).Data;
 distHandle =  block.InputPort(12).Data;
% v_mpc = block.InputPort(7).Data;
% uLastMove = block.InputPort(8).Data;

mpcverbosity off
% MPC object setup
mpcRight=block.DialogPrm(1).Data ;   %Sampling time
xmpcRight = block.DialogPrm(2).Data ;   %Sampling time
mpcLeft=block.DialogPrm(3).Data ;   %Sampling time
xmpcLeft = block.DialogPrm(4).Data ;   %Sampling time
mpcTip=block.DialogPrm(5).Data ;   %Sampling time
xmpcTip = block.DialogPrm(6).Data ;   %Sampling time
```

```
all = [1 2 3 4 5 6];
size(uLast);
for iter = 1:maxIter
%define disturbance
for i = 1:size(uOptLast,2)+1
   vDist(i) = v;
         pDist(:,i) = pdm;
   if distHandle == 0
      % assume constant disturbance

      uLastDist(:,i) = uLast;
   else
      %assume last optimization
      if i == 1
         uLastDist(:,i) = uLast;
      else
         uLastDist(:,i) = uOptLast(:,i-1);
      end
   end
end

% right engine mpc

mvRight = kRight;
mdRight = setdiff(all,mvRight);

InputGroupRight=struct('Manipulated',[mvRight],...
   'Measured',[mdRight 7:17]);
mpcRight.Model.Plant.InputGroup = InputGroupRight;



%disturbRight = [uLast(mdr,:)' v' pdm'];
disturbRight = [uLastDist(mdRight,:)' vDist' pDist'];

xmpcRight.LastMove = uLast(mvRight,1);
% xmpcRight.Plant = [x;uLast];
ManipulatedVariables(1) = struct('RateMin',-10,'RateMax',10);
if mvRight == 1
   OutputVariables(9) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(10) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(11) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(12) = struct('Min',0.75,'Max',Inf);
elseif mvRight == 2
   OutputVariables(9) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(10) = struct('Min',-Inf,'Max',Inf);
```

```
   OutputVariables(11) = struct('Min',0.75,'Max',Inf);
   OutputVariables(12) = struct('Min',0.75,'Max',Inf);
elseif mvRight ==3
   OutputVariables(9) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(10) = struct('Min',0.75,'Max',Inf);
   OutputVariables(11) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(12) = struct('Min',-Inf,'Max',Inf);
elseif mvRight ==4
   OutputVariables(9) = struct('Min',0.75,'Max',Inf);
   OutputVariables(10) = struct('Min',0.75,'Max',Inf);
   OutputVariables(11) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(12) = struct('Min',-Inf,'Max',Inf);
elseif mvRight ==5
   OutputVariables(9) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(10) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(11) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(12) = struct('Min',-Inf,'Max',Inf);
else
   OutputVariables(9) = struct('Min',0.75,'Max',Inf);
   OutputVariables(10) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(11) = struct('Min',0.75,'Max',Inf);
   OutputVariables(12) = struct('Min',-Inf,'Max',Inf);
end
% set(mpcRight,'ManipulatedVariables',ManipulatedVariables);
% set(mpcRight,'OutputVariables',OutputVariables);


 [uRight,InfoRight]=mpcmove(mpcRight,xmpcRight,y,r(1,:),disturbRight);

 % Left engine mpc

mvLeft = kLeft;
mdLeft = setdiff(all,mvLeft);

InputGroupLeft=struct('Manipulated',[mvLeft],...
   'Measured',[mdLeft 7:17]);
mpcLeft.Model.Plant.InputGroup = InputGroupLeft;


%disturbLeft = [uLast(mdr,:)' v' pdm'];
disturbLeft = [uLastDist(mdLeft,:)' vDist' pDist'];

xmpcLeft.LastMove = uLast(mvLeft,1);
% xmpcLeft.Plant = [x;uLast];
ManipulatedVariables(1) = struct('RateMin',-10,'RateMax',10);
if mvLeft == 1
```

```
    OutputVariables(9) = struct('Min',-Inf,'Max',Inf);
    OutputVariables(10) = struct('Min',-Inf,'Max',Inf);
    OutputVariables(11) = struct('Min',-Inf,'Max',Inf);
    OutputVariables(12) = struct('Min',0.75,'Max',Inf);
elseif mvLeft == 2
    OutputVariables(9) = struct('Min',-Inf,'Max',Inf);
    OutputVariables(10) = struct('Min',-Inf,'Max',Inf);
    OutputVariables(11) = struct('Min',0.75,'Max',Inf);
    OutputVariables(12) = struct('Min',0.75,'Max',Inf);
elseif mvLeft ==3
    OutputVariables(9) = struct('Min',-Inf,'Max',Inf);
    OutputVariables(10) = struct('Min',0.75,'Max',Inf);
    OutputVariables(11) = struct('Min',-Inf,'Max',Inf);
    OutputVariables(12) = struct('Min',-Inf,'Max',Inf);
elseif mvLeft ==4
    OutputVariables(9) = struct('Min',0.75,'Max',Inf);
    OutputVariables(10) = struct('Min',0.75,'Max',Inf);
    OutputVariables(11) = struct('Min',-Inf,'Max',Inf);
    OutputVariables(12) = struct('Min',-Inf,'Max',Inf);
elseif mvLeft ==5
    OutputVariables(9) = struct('Min',-Inf,'Max',Inf);
    OutputVariables(10) = struct('Min',-Inf,'Max',Inf);
    OutputVariables(11) = struct('Min',-Inf,'Max',Inf);
    OutputVariables(12) = struct('Min',-Inf,'Max',Inf);
else
    OutputVariables(9) = struct('Min',0.75,'Max',Inf);
    OutputVariables(10) = struct('Min',-Inf,'Max',Inf);
    OutputVariables(11) = struct('Min',0.75,'Max',Inf);
    OutputVariables(12) = struct('Min',-Inf,'Max',Inf);
end
% set(mpcLeft,'ManipulatedVariables',ManipulatedVariables);
% set(mpcLeft,'OutputVariables',OutputVariables);
 [uLeft,InfoLeft]=mpcmove(mpcLeft,xmpcLeft,y,r(1,:),disturbLeft);

 % tip-jet engine mpc

mvTip = kTip;
mdTip = setdiff(all,mvTip);

InputGroupTip=struct('Manipulated',[mvTip],...
    'Measured',[mdTip 7:17]);
mpcTip.Model.Plant.InputGroup = InputGroupTip;
```

```
%disturbTip = [uLast(mdr,:)' v' pdm'];
disturbTip = [uLastDist(mdTip,:)' vDist' pDist'];

xmpcTip.LastMove = uLast(mvTip,1);
% xmpcTip.Plant = [x;uLast];
ManipulatedVariables(1) = struct('RateMin',-10,'RateMax',10);
% ManipulatedVariables(1)=struct('RateMin',-0.0005,'RateMax',0.0005);
if mvTip == 1
   OutputVariables(9) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(10) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(11) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(12) = struct('Min',0.75,'Max',Inf);
elseif mvTip == 2
   OutputVariables(9) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(10) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(11) = struct('Min',0.75,'Max',Inf);
   OutputVariables(12) = struct('Min',0.75,'Max',Inf);
elseif mvTip ==3
   OutputVariables(9) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(10) = struct('Min',0.75,'Max',Inf);
   OutputVariables(11) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(12) = struct('Min',-Inf,'Max',Inf);
elseif mvTip ==4
   OutputVariables(9) = struct('Min',0.75,'Max',Inf);
   OutputVariables(10) = struct('Min',0.75,'Max',Inf);
   OutputVariables(11) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(12) = struct('Min',-Inf,'Max',Inf);
elseif mvTip ==5
   OutputVariables(9) = struct('Min',0.75,'Max',Inf);
   OutputVariables(10) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(11) = struct('Min',0.75,'Max',Inf);
   OutputVariables(12) = struct('Min',-Inf,'Max',Inf);
else
   OutputVariables(9) = struct('Min',0.75,'Max',Inf);
   OutputVariables(10) = struct('Min',-Inf,'Max',Inf);
   OutputVariables(11) = struct('Min',0.75,'Max',Inf);
   OutputVariables(12) = struct('Min',-Inf,'Max',Inf);
end
% set(mpcTip,'ManipulatedVariables',ManipulatedVariables);
% set(mpcTip,'OutputVariables',OutputVariables);
 [uTip,InfoTip]=mpcmove(mpcTip,xmpcTip,y,r(1,:),disturbTip);



 uOptLast(mvRight,:) = InfoRight.Uopt';
 uOptLast(mvLeft,:) = InfoLeft.Uopt';
```

```
uOptLast(mvTip,:) = InfoTip.Uopt';
%iter = iter +1
end

block.OutputPort(1).Data = uOptLast(:,1);
block.OutputPort(2).Data = uOptLast;;
block.OutputPort(3).Data = InfoTip.Yopt(1,:)';
```

REFERENCES

ADIBHATLA, S. "Propulsion Control Law Design for the NASA STOL Controls Technology Program." *AIAA International Powered Lift Conference.* Santa Clara, CA, 1993. 1-11.

ADIBHATLA, S., and Z. GASTINEAU. "Tracking Filter Selection and Control Model Selection for Model Based Control." *30th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.* Indianapolis, IN, 1994.

ADIBHATLA, S., G.J. COLLIER, and S. GARG. "H-Infinity Control Design for a Jet Engine." *34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit.* Cleveland, OH: AIAA, 1998. 1-11.

ALLGOWER, F., and A. ZHENG. *Nonlinear Model Predictive Control.* Birkhauser, 2000.

ARULAMPALAM, M. S., S. MASKELL, N. GORDON, and T. CLAPP. "A Turtorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking." *iEEE Transactions on Signal Processing* 50, no. 2 (2002): 174-188.

BALLIN, M. G. "A High Fidelity Real-Time Simulation of a Small Turboshaft Engine." NASA/TM-1988-100991, 1988.

BEMPORAND, A., M. MORARI, and N. L. RICKER. "Model Predictive Control Tookbox 2 - User's Guide." Matlab, 2007.

BEMPORAND, A., M. MORARI, V. DUA, and E. N. PISTIKOPOULOS. "The Explicit Linear Quadratic Regulator for Constrained Systems." *Automatica* 38 (2002): 3-20.

BORGUET, S., and O LEONARD. "The Fisher Information Matrix as a Relevant Tool for Sensor Selection in Engine Health Monitoring." *International Journal of Rotating Machinary*, 2008: 1-13.

BRISTOL, E. "On a New Measure of Inteaction for Multivariable Process Control." *IEEE Transactions of Automatic Control* 11, no. 1 (1966): 133-134.

BROWN, H., and J. A. ELGIN. "Aircraft Engine Control Mode Analysis." *Journal of Engineering for Gas Turbines and Power* 107 (1985): 838-844.

BRUNELL, B. J., D. E. VIASSOLO, and R. PRASANTH. "Model Adaptation and Nonlinear Model Predictive Control of an Aircraft Engine." *Proceedings of ASME Turbo Expo 2004 Power for Land, Sea, and Air.* Vienna, Austria, 2004. 673-682.

BRUNELL, B., R. R. BITMEAD, and A. J. CONNOLLY. "Nonlinear Model Predictive Control of an Aircraft Gas Turbine Engine." *Proceedings of the 41st IEEE Conference on Decision and Control.* Las Vegas, Nevada, 2002.

CAMACHO, E. F., and C. BORDONS. *Model Predictive Control.* Springer, 2004.

CHATTERJEE, S., and J. S. LITT. "Online Model Parameter Estimation of Jet Engine Degradation for Autonomous Propulsion Control." NASA/TM-2003-212608, 2003.

CHUNG, K., K. R. LEAMY, and T. P. COLLINS. "A Turbine Engine Aerodynamic Model for In-Stall Transient Simulation." AIAA-85-142, 1985.

DADD, G. J., A. E. SUTTON, and A. W.M. GRIEG. "Multivariable Control of Military Engines." *AGARD PEP Symposium on "Advanced Aero-Engine Concepts and Controls".* Seattle, WA, 1995.

D'AMATO, F. J. "Industrial Application of a Model Predictive Control Solution for Power Plant Startups." *International Conference on Control Applications.* Munich, Germany, 2006.

DUNBAR, W. B. "Distributed Receding Horizon Control of Dynamically Coupled Nonlinear Systems." *IEEE Transactions on Automatic Control* 52, no. 7 (2007): 1249-1263.

EDMUNDS, J. M. "Control System Design and Analysis Using Closed-Loop Nyquist and Bode Arrays." *International Journal of Control* 30 (1979): 773-802.

FREDERICK, D. K., S. GARG, and T. P. COLLINS. "Turbofan Engine Control Design Using Robust Multivariable Control Technologies." *IEEE Transactions on Control Systems Technology* 8, no. 6 (November 2000): 961-970.

FULLER, J. W. Square Root Method For Computationally Efficient Model Predictive Control. USA Patent 7197485 B2. March 27, 2007.

FULLER, J. W., A. KUMAR, and R. C. MILLER. "Adaptive Model Based Control of Aircraft Propulsion System." *Proceedings of ASME Turbo Expo 2006: Power for Land, Sea, and Air.* Barcelona, Spain, 2006.

FULLER, J., D. SETO, and R. MEISNER. "Optimization-Based Control for Flight Vehicles." *AIAA Guidance, Navigation, and Control Conference.* Denver, CO: AIAA 2000-4055, 2000.

GARG, S. "Turbofan Engine Control System Design Using the LQG/LTR Methodology." *American Controls Conference Proceedings.* 1989. 134-141.

GARG, S., D.L. MATTERN, M. M. BRIGHT, and P. J. OUZTS. "H-Infinity Based Integrated Flight/Propulsion Control Design for a STOVL Aircraft in Transition Flight." *Guidance, Navigation, and Control Conference.* Portland, Oregon, 1990. 1-30.

GUSTAFSSON, F. *Adaptive Filtering and Change Detection.* New York: John Wiley and Sons, LTD, 2000.

HOSNY, W. M., and S. J. BITTER. "Turbofan Engine Nonrecoverable Stall Computer Simulation Development and Validation." AIAA-85-143, 1985.

JAW, L., and S. GARG. "Propulsion Control Technology Development in the United States: A Historical Perspective." NASA/TM-20050213978, 2005.

KEVICZKY, T., F. BORRELLI, and G. J. BALAS. "A Study on Decentralized Receding Horizon Control for Decoupled System." *Proceedings of the 2004 American Control Conference.* Boston, MA, 2004.

KHALID, S. J., and R. E. HEARNE. "Enhancing Dynamic Model Fidelity for Improved Prediction of Turbofan Engine Transient Performance." AIAA-80-1083, 1980.

KOBAYASHI, T., D. L. SIMON, and J. S. LITT. "Application of Constant Gain Extended Kalman Filter for In-Flight Estimation of Aircraft Engine Performance." NASA/TM-2005-213865, 2005.

KONG, C., J. PARK, and M. KANG. "A Study on Transient Performance Characteristics of the Canard Rotor Wing Type Unmanned Aerial Vehicle Propulsion System During Flight Mode Transition." *Journal of Engineering for Gas Turbines and Power* 128 (July 2006): 573-578.

KOPASAKIS, G., J. W. CONNOLLY, D. E. PAXSON, and P. MA. *Volume Dynamics Propulsion System Modeling for Supersonics Vehicle Research.* NASA/TM-2008-215172, 2008.

KOTHARE, S., and M. MORARI. "Contractive Model Predictive Control for Constrained Nonlinear Systems." *IEEE Transactions on Automatic Control* 45, no. 6 (2000): 1053-1071.

KRISHNAKUMAR, K., S. NARAYANSWAMY, and S. GARG. "Integrator Windup Protection - Techniques and a STOVL Aircraft Engine Controller Application." *AIAA, Guidance, Navigation, and Control Conference.* San Diego, CA: AIAA, 1996. 1-14.

LING, K. V., J. M. MACIEJOWSKI, and B. F. WI. "Multiplexed Model Predictive Coontrol." *16th IFAC World Congress.* 2005.

LITT, J. S., et al. "A Survey of Intelligent Control and Health Management Technologies for Aircraft Propulsion Systems." *Journal of Aerospace Computing, Information, and Communication* 1 (2004): 543-563.

LUPPOLD, R. H., J. R. ROMAN, G. W. GALLOPS, and L. J. KERR. "Estimating In-Flight Engine Performance Variations Using Kalman Filter Concepts." *AIAA/ASME SAE/ASEE 25th Joing Propulsion Conference.* Monterey, CA, 1989. 1-8.

LYTLE, J. K. "The Numerical Propulsion System Simulation: An Overview." NASA/TM-2000-209915, 2000.

MACIEJOWSKI, J. M. *Multivariable Feedback Design.* Addison-Wesley, 1989.

—. *Predictive Control With Constraints.* Prentice Hall, 2002.

MAVRIS, D. N., J. TAI, and D. P. SCHRAGE. "A Multidisciplinary Design Optimization Approach to Sizing Stopped Rotor Configurations Utilizing Reaction Drive and Circulation Control." AIAA-94-4296, 1994.

MCKENNA, J. T. "One Step Beyond." *Rotor & Wing.* February 2007.

MITCHELL, C. A., and B. J. VOGEL. "The Canard Rotor Wing (CRW) Aircraft - A New Way to Fly." *AIAA International Air and Space Symposium and Exposition: The Next 100 Years.* Dayton, OH: AIAA 2003-2517, 2003.

MU, J., D. REES, and G. P. LIU. "Approximate Model Predictive Control for Gas Turbine Engines." *Proceedings of the 2004 American Control Conference.* Boston, MA, 2004.

MUSKE, K. R., and T. A. BADGWELL. "Disturbance Modeling For Offset-Free Linear Model Predictive Control." *Journal of Process Control*, 2002: 617-632.

POLLEY, J. A., S. ADIBHATLA, and P. J. HOFFMAN. "Multivariable Turbofan Engine Control for Full Flight Envelope Operation." *Journal of Engineering for Gas Turbines and Power* 111 (1989): 130-137.

QIN, S. J., and T. A. BADGWELL. "A Survey of Industrial Model Predictive Control Technology." *Control Engineering Practice* 11, no. 7 (2003): 733-764.

RAO, C. V., S. J. WRIGHT, and J. B. RAWLINGS. "Application of Interior-Point Methods to Model Predictive Control." *Journal of Optimization Theory and Applications* 99, no. 3 (1998): 723-757.

"Real-Time Modeling Methods for Gas Turbine Engine Performance." Aerospace Information Report, Society of Automotive Engineers, Inc., July 2001.

RICHARDS, A., and J. P. HOW. "Robust Distributed Model Predictive Control." *International Journal of Control* 80, no. 9 (2007): 1517-1531.

RICHTER, H., A. SINGARAJU, and J. LITT. "Multiplexed Predictive Control of a Large Commercial Turbofan Engine." *Journal of Guidance, Control, and Dynamics* 16 (2008): 273-281.

SANGHI, V., B. K. LAKSHMANAN, and V. SUNDARARAJAN. "Survey of Advancements in Jet-Engine Thermodynamic Simulation." *Journal of Propulsion and Power* 16 (2000): 797-807.

SCOKAERT, P. O.M., D. Q. MAYNE, and J. B. RAWLINGS. "Suboptimal Model Predictive Control (Feasibility Implies Stability)." *IEEE Transactions on Automatic Control* 44, no. 3 (1999): 648-654.

SKIRA, C., and R. DEHOFF. "A Practical Approach to Linear Model Analysis for Multivariable Turbine Engine Control Design." *Alternatives for Linear Multivariable Control, National Engineering Consortium, Inc.* Chicago, 1978.

SKOGESTAD, S., and I. POSTLETHWAITE. *Multivariable Feedback Control: Analysis and Design, 2nd Ed.* New York: Wiley, 2005.

SPANG, H. A., and H. BROWN. "Control Of Jet Engines." *Control Engineering Practice 7*, 1999: 1043-1059.

STRANG, G. *Linear Algebra and its Applications, Fourth Edition.* Belmont, CA: Thomson Brooks/Cole, 2006.

TAI, J. C.M. "A Multidisciplinary Design Approach to Size Stopped Rotor/Wing Configurations Using Reaction Drive and Circulation Control." PhD Dissertation, Georgia Institute of Technology, Atlanta, GA, 1998.

TALBOT, P. D. "High Speed Rotorcraft: Comparison of Leading Concepts and Technology Needs." *47th Annual Forum of the American Helicoptor Society.* Phoenix, AZ, 1991. 1187-1212.

TRODDEN, P., and A. RICHARDS. "Robust Distributed Model Predictive Control Using Tubes." *Proceedings of the 2006 American Control Conference.* Minneapolis, Minnesota, 2006.

VAN ESSEN, H. A., and H. C. DE LANGE. "Nonlinear Model Predictive Control Experiments on a Laboratory Gas Turbine Installation." *Journal of Engineering for Gas Turbines and Power* 123 (2001): 347-352.

VENKAT, A.N. *Distributed Model Predictive Control: Theory and Applications.* PhD Dissertation, University of Wisconsin-Madison, 2006.

VIASSOLO, D., A. KUMAR, and B. BRUNELL. "Advanced Control for Fuel Consumption and Time-On-Wing Optimization in Commercial Aircraft Engine." *Proceedings of GT2007 ASME Turbo Expo 2007: Power for Land, Sea, and Air.* Montreal, Canada, 2007.

VOLPONI, A. "Enahanced Self tuning On-Board Real-Time Model (eSTORM) for Aircraft Engine Performance Health Tracking." NASA/CR-2008-215272, 2008.

WATTS, S. R., and S. GARG. "A Comparison of Multivariable Control Design Techniques for a Turbofan Engine Control." *40th Gas Turbine and Aeroengine Congress and Exposition.* Houston, TX, 1995.

WRIGHT, S. J. "Applying New Optimization Algorithms To Model Predictive Control." *Chemical Process Control-V* 93 (1997): 147-155.